Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія розпізнавання жестів для керування у комп'ютерних іграх»

Виконав: студент 2-го курсу, групи 1КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Шевчук А. В.
(прізвище та ініціали)

Керівник: к. т. н., доцент каф. КН
Арсенюк І. Р.
(прізвище та ініціали)
«15» 12 2022 р.

Опонент: к. т. н., доцент каф. АІІТ
Маслій Р. В.
(прізвище та ініціали)
«15» 12 2022 р.

Допущено до захисту
Завідувач кафедри КН
д. т. н., проф. Яровий А. А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
14 вересня 2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Шевчука Андрія Віталійовича

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка інформаційної технології розпізнавання жестів для керування у комп'ютерних іграх.
   керівник роботи: к. т. н., доцент кафедри КН Арсенюк І. Р.
   затверджені наказом вищого навчального закладу від «14» 09 2022 року № 203
2. Строки подання студентом роботи 18 листопада 2022 року
3. Вихідні дані до роботи:
   Мінімальна кількість жестів для розпізнання – 6 од., мінімальна точність розпізнавання жестів – 95%, рекомендована розмірність відеоряду – 1280×720 пікселів, середовище розробки із підтримкою веб-камери, об'єктно-орієнтована мова програмування.
4. Зміст текстової частини:
   Вступ, аналіз предметної області, аналіз існуючих рішень, обґрунтування методу розв'язання задачі, обґрунтування вибору інструментів технічної реалізації, проектування інформаційної технології розпізнавання жестів для керування комп'ютерними іграми, програмна реалізація інформаційної технології розпізнавання жестів для керування комп'ютерними іграми, аналіз результатів тестування, економічна частина, висновок, розробка інструкції користувача.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
   Загальна структурна схема інформаційної технології, діаграма класів інформаційної технології, схема алгоритму роботи технології розпізнавання жестів для керування у комп'ютерних іграх, приклад роботи нейронної мережі.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Арсенюк І.Р., к.т.н., доц. каф. КН	14.09.2022	19.11.2022
4	Нікіфорова Л. О., к. е. н., доц. каф. ЕПВМ	19.09.2022	12.12.2022

7. Дата видачі завдання 14 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного стану розвитку систем розпізнавання жестів для керування у комп'ютерних іграх. Постановка задач дослідження	14.09.2022	01.10.2022	Аналітичний огляд літературних джерел, задачі досліджень, розділ 1
2	Обгрунтування методу розв'язання задачі, обгрунтування вибору інструментів технічної реалізації	02.10.2022	16.10.2022	Моделі, розділ 2
3	Проектування інформаційної технології. Програмна реалізація інформаційної технології. Аналіз результатів тестування	17.10.2022	7.11.2022	розділ 3
4	Підготовка економічної частини	08.11.2022	21.11.2022	розділ 4
5	Апробація та/або впровадження результатів дослідження	23.11.2022	01.12.2012	тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.11.2022	14.12.2012	Пояснювальна записка, графічний матеріал, презентація

Студент _____ (підпис) Шевчук А.В.

Керівник роботи _____ (підпис) Арсенюк І.Р.

АНОТАЦІЯ

УДК 004.8

Шевчук А.В. Інформаційна технологія розпізнавання жестів для керування у комп'ютерних іграх. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма – комп'ютерні науки. Вінниця: ВНТУ, 2022. 109 с.

На укр. мові. Бібліогр.: 35 назв; рис.: 31; табл. 19.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології розпізнавання жестів для керування у комп'ютених іграх. У роботі досліджено відомі продукти, що надають можливість здійснювати керування у комп'ютерних іграх засобами розпізнавання жестів, визначено їх основні недоліки. Розроблено структурну схему алгоритму роботи системи розпізнавання жестів для керування у комп'ютерних іграх. Обрано інструменти для програмної реалізації інформаційної системи. Здійснено реалізацію засобами мови програмування Python та бібліотеки для машинного навчання Tensorflow. Виконано тестування розробленого програмного забезпечення, яке показало правильність його функціонування. Точність розпізнавання склала 95,5%.

Графічна частина складається з 5 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 776585 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника склав 0,83 роки та економічний ефект для споживача при використанні даної розробки.

Ключові слова: жестове керування, розпізнавання рухів, керування у комп'ютерних іграх.

ABSTRACT

Shevchuk A.V. Gesture recognition information technology for control in computer games. Master's qualification thesis on specialty 122 – computer science, educational program – computer science. Vinnytsia: VNTU, 2022. 109 p.

In Ukrainian languages Bibliography: 35 titles; Fig.: 31; table 19.

This master's thesis is devoted to the development of gesture recognition software for controlling computer games. The work examines well-known products that provide an opportunity to control computer games with gesture recognition, and identifies their main shortcomings. A structural diagram of the algorithm of the gesture recognition system for controlling computer games has been developed. The tools for software implementation of the information system were chosen. Real-world use of Python programming language and Tensorflow machine learning library. Testing of the developed software was conducted, which showed the correctness of its functioning.

The graphic part consists of 5 posters.

In the economic section, the amount of costs for the development and production of a new technical solution is calculated, which is 776585 hryvnias, the estimated amount of costs for each of the cost items is predicted, the net profit is calculated, the payback period for the manufacturer is 0,83 years and the economic effect for consumption when applying this development.

Keywords: gesture control, motion recognition, computer game controls.

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ СИСТЕМ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ КЕРУВАННЯ КОМП'ЮТЕРНИМИ ІГРАМИ 10
1.1 Огляд проблем в області розпізнавання жестів для керування комп'ютерними іграми 11
1.2 Аналіз існуючих рішень і аналогів 13
1.3 Постановка задачі і формулювання вимог до інтелектуальної системи розпізнавання жестів для керування комп'ютерними іграми 22
1.4 Висновок до розділу 1 24
2 ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ КЕРУВАННЯ У КОМП'ЮТЕРНИХ ІГРАХ 24
2.1Варіантний аналіз шляхів розв'язання поставленої задачі 24
2.2 Обгрунтування вибору моделей навчання нейронних мереж 29
2.3 Розробка методу попередньої обробки зображення 36
2.4 Розробка загальної структурної схеми функціонування інтелектуальної системи розпізнавання жестів для керування у комп'ютерних іграх 42
2.5 Висновок до розділу 2 44
3 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ КЕРУВАННЯ У КОМП'ЮТЕРНИХ ІГРАХ 45
3.1 Обґрунтування вибору інструментів технічної реалізації 45
3.2 Розробка загального алгоритму розпізнавання жестів для керування комп'ютерними іграми 51
3.3 Розробка UML-діаграми взаємодії компонентів системи 53
3.4 Розробка програмного забезпечення 55
3.5 Тестування розробленої системи і аналіз результатів її роботи 61
3.6 Висновок до розділу 3 71
4 ЕКОНОМІЧНА ЧАСТИНА 72
4.1 Комерційний та технологічний аудит науково-технічної розробки 72

4.2 Прогнозування витрат на виконання науково-дослідної та дослідно-конструкторської роботи 75
4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором 81
4.4 Висновок до розділу 4 86
ВИСНОВКИ 87
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 89
ДОДАТОК А (обов'язковий) Результат перевірки на плагіат в онлайн-системі UNICHECK 94
ДОДАТОК Б (обов'язковий) Лістинг програми 95
ДОДАТОК В (обов'язковий) Графічна частина 104

ВСТУП

Актуальність теми дослідження. Багато чим сучасні технології комп'ютерної інженерії завдячують розвитку індустрії комп'ютерних ігор. Ще в часи, коли персональні комп'ютери тільки з'являлись, паралельно з ними з'являлись і комп'ютерні ігри. І частіше за все саме ігри диктували вимоги з апаратного забезпечення, адже вимагали високої обчислювальної потужності. Графічні та звукові карти, процесори, CD, DVD та флеш-носії, а також приводи для зчитування та опрацювання даних на цих носіях – все це було в першу чергу потребою розробників ігор.

Комп'ютерна гра – програма, призначена для організації ігрового процесу, зв'язку з ігровим середовищем і його складовими. Поняття «комп'ютерна гра» стосується будь-яких обчислювальних машин (комп'ютерів), не варто плутати його з поняттям «PC-гра». Адже останнє означає, що гра призначена саме для персональних комп'ютерів, в той час як комп'ютерні ігри випускають для мобільних пристроїв, ігрових приставок, грального автоматів, а деякі умільці навіть запускають ігри на калькуляторах, терміналах чи банкоматах, загалом на будь-яких обчислювальних пристроях.

Комп'ютерні ігри можуть бути як альтернативним способом відпочинку, так і спортивними дисциплінами. Люди люблять ігри, а компанії вкладають величезні кошти для того, щоб зробити ігровий процес цікавим і урізноманітнити його. Ще 20 років у суспільстві переважала думка, що комп'ютерні ігри можуть цікавити лише дітей, та насправді серед гравців як тоді, так і зараз були і є люди різних вікових категорій, соціальних груп, родів діяльності і географічних належностей. Ігри об'єднують людей, індустрія розвивається, фанатів стає все більше. За підрахунками аналітиків кількість гравців уже досягла позначки у 3 мільярди [1].

Комп'ютерні ігри у своїй суті часто симулюють обставини і ситуації, схожі або не схожі до обставин і ситуацій з реального життя. Завдяки самій їх ідеї зараз людство має багато симуляторів, які допомагають навчанню у медицині, сільському господарстві, військовій справі тощо. Людство розвивається, а

симулятори значно покращують і пришвидшують процес навчання, дозволяючи не витрачати додаткових ресурсів. Хірурги практикують операції, водії і пілоти покращують власне розуміння поведінки машин. Оператори військових розвідувальних дронів і інших безпілотних літальних апаратів для віддаленого керування використовують джойстики, за принципом дії аналогічні до джойстиків від ігрових приставок.

У пошуках способів дати гравцеві новий ігровий досвід виникли ідеї про залучення до керування не лише миші, клавіатури чи геймпада, а й самого гравця, фізично. Чим більше участі в процесі бере гравець – тим цікавіше йому грати. Спочатку з’явились спеціальні бездротові контролери для ігрових приставок, які користувач тримав у руках. Завдяки ним система легко отримувала інформацію про положення рук гравця, відстежуючи лише контролери. З часом ринок наповнився великою кількістю різних пристроїв, які працюють за схожим сценарієм. Однак серед них є і інші технічні рішення для залучення користувачів до нової взаємодії з грою. Наприклад, шоломи віртуальної реальності або системи визначення положення тіла людини у тривимірному просторі засобами лише камери і кількох додаткових датчиків глибини. Кілька таких сміливих проектів були запущені у продаж, однак майже жоден з них не знайшов достатнього об’єму аудиторії, що зробило виробництво економічно невигідним, а тому їх історії закінчились доволі швидко.

Отже, переважна більшість існуючих систем розпізнавання жестів і рухів для керування грою потребують додаткового апаратного забезпечення у процесі жестового керування задля компенсації низької точності розпізнавання жестів і рухів людини. Ці аксесуари означають додаткові фінансові витрати для користувача. Тому розробка програмного продукту, який дозволить досягнути високого рівня точності і швидкості розпізнавання без залучення аксесуарів з датчиками, використовуючи лише відеопотік з веб-камери, є актуальною.

Метою дослідження магістерської кваліфікаційної роботи є підвищення точності розпізнавання жестів у неперервному відеопотоці даних за допомогою розробки інформаційної технології ідентифікації візуальних команд учасника

комп’ютерної гри, що дозволить уникнути застосування додаткових аксесуарів у процесі керування комп’ютерними іграми. Для досягнення поставленої мети необхідно розв’язати наступні завдання:

− розглянути та проаналізувати існуючі реалізації розв’язання задачі розпізнавання жестів;

− запропонувати метод покращення процесу розпізнавання для інформаційної технології розпізнавання жестів;

− запропонувати етапи інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;

− виконати програмну реалізацію запропонованої інформаційної технології розпізнавання жестів;

− здійснити тестування програмного продукту та виконати аналіз отриманих результатів;

− виконати комерційний та технологічний аудит розробленого продукту та обґрунтувати економічну доцільність розробки інформаційної технології розпізнавання жестів для керування комп’ютерними іграми.

Об’єктом дослідження є процес розпізнавання жестів та застосування їх у керуванні комп’ютерними іграми.

Предметом дослідження є інформаційна технологія розпізнавання жестів.

Методи дослідження – методи розпізнавання образів, теорія нейронних мереж, методи масштабування та інтеграції програмного забезпечення, методи та підходи до розробки систем штучного інтелекту, методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному: удосконалено процес обробки зображень шляхом використання моделі поєднання методу інтегрального представлення зображення та нейронної мережі, що дозволило підвищити точність розпізнавання жестів і підвищити якість жестового керування в іграх без залучення додаткових периферійних засобів.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено алгоритм розпізнавання жестів з використанням нейронної мережі на основі ознак Хаара.

2. Розроблено програмний продукт розпізнавання жестів з використанням нейронної мережі на основі ознак Хаара.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. Здобувачу належить праці, написана у співавторстві: «Застосування інформаційної технології розпізнавання жестів для керування у комп'ютерних іграх».

Апробація результатів роботи. Результати досліджень апробовані на III міжнародній науково-практичній конференції «Scientific Progress: Innovations, Achievements And Prospects».

Публікації. За результатами досліджень опубліковано матеріали на III міжнародній науково-практичній конференції «Scientific Progress: Innovations, Achievements And Prospects» [1].

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ СИСТЕМ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ КЕРУВАННЯ КОМП'ЮТЕРНИМИ ІГРАМИ

1.1 Аналіз предметної області розпізнавання жестів

Існує багато проблем у реалізації технології розпізнавання жестів, у тому числі й соціальні. Жести мають бути не лише простими у виконанні, але й зрозумілими, що зробить їх використання інтуїтивно простим, універсальними. Розпізнавання жестів дозволяє комп'ютерам бути доступнішими для людей з обмеженими фізичними можливостями і робить взаємодію людини з технікою більш природньою в іграх та віртуальному світі. Прикладом простого використання методу розпізнавання жестів є заміна курсора, адже можна вказувати пальцем на екран комп'ютера, а курсор буде відповідно переміщуватись. Потенційно це зробить непотрібним такі прилади, як мишка, клавіатура і навіть сенсорні екрани.

Кілька років тому Google AI опублікували підхід для розпізнавання жестів у реальному часі з камери телефона. Модель реалізована у відкритому фреймворку MediaPipe, що призначений для обробки відео та аудіо. Переважна більшість сучасних рішень потребує великої обчислювальної потужності комп'ютера, а підхід від Google, навідміну від них, видає результат в реальному часі на телефоні.

Іншим прикладом вдалого впровадження розпізнавання жестів є системи, інтегровані в сервери презентацій. Це значно розширює можливості презентаційного обладнання. Такі системи призначені для обладнання конференц-залів, залів засідань, навчальних аудиторій тощо. Будучи частиною комплексу презентаційного обладнання, система розпізнавання жестів дозволяє спростити проведення презентації і зробити його максимально зручним як для ведучого, так і для аудиторії. За допомогою відеокамери та спеціальних сенсорів глибини система розпізнає жести людини і дає можливість керувати показом без використання

пристроїв. Система розпізнає жести в тривимірній проекції, завдяки чому можна не тільки перегортати слайди презентації, але також наближати і віддаляти зображення, повертати і розтягувати на екрані 3D-об'єкти, керувати налаштуваннями аудіо і відео.

Варті уваги також розумні колонки, які спочатку були спроектовані для управління за допомогою голосових команд. Вони не лише можуть виконувати задані функції, а й підтримувати діалог з користувачем, сповіщаючи про результати пошуку в мережі Інтернет або уточнюючи, чого саме він хоче. Нещодавно опублікований патент компанії Apple свідчить про те, що компанія вивчає інші методи введення інформації, які не пов'язані ні з голосом, ні з сенсорними екранами. Мається на увазі управління на базі системи жестів. У патенті йдеться про те, що розумна колонка буде реагувати на рухи рук, махи і хлопки. Тим часом в нових поколіннях розумних колонок Amazon Echo та Яндекс-станції уже застосовується альтернативне керування за допомогою жестів. Рухом руки можна відрегулювати гучність, вимкнути будильник або зупинити відтворення аудіо чи відео.

Розпізнавання жестів знайшло застосування і в автомобільній індустрії. Система CarPlay та її аналоги покликані полегшити користування мультимедійними пристроями в автомобілі. За статистикою американської автомобільної асоціації, близько 58% дорожньо-транспортних пригод стаються через неуважність водіїв, з них більшість – через відволікання від дороги [2]. Взаємодія з мультимедійними екранами під час руху спричиняє загрозу. Системи жестового керування вирішують цю проблему: водій, не відводячи очей від дороги, може виконати кілька рухів рукою перед екраном замість натискання на кнопки і прийняти дзвінок, переключити пісню, змінити відображувану на екрані інформацію тощо.

Технології розпізнавання мають безмежний потенціал. Це перспективна область науки, що стрімко розвивається, тому зчитування і обробка рухів людини знаходять велику кількість різноманітних способів і галузей застосування.

1.2 Аналіз існуючих рішень і аналогів

Розглянемо відомі розробки зі втіленням керування комп’ютерними програмами на основі рухів і жестів людини.

The Leap – це невеликий USB-пристрій, розроблений для користувачів, робочою частиною розташовується вгору, тим самим створюючи 3D-область взаємодії об'ємом близько 227 кубічних дециметрів (Рис. 1.1 – периферійний пристрій розпізнавання жестів). The Leap відстежує рух пальців і рук, олівців, ручок, паличок для їжі з великою точністю [3].

Рисунок 1.1 – Периферійний пристрій візуального розпізнавання жестів

Myo Gesture Control – браслет для руки, складається з 8 секцій, що зчитують електричну активність імпульсів, що виникають при скороченні м'язів, щоб надати можливість управляти технологією за допомогою рухів руки (Рис. 1.2 – Периферійний пристрій відстеження м’язової активності). Для визначення активності браслета на ньому розташований світлодіод, який є індикатором. Секції

з'єднуються еластичними кріпленнями, що дозволяють використовувати гаджет на різну ширину передпліччя [4].

Рисунок 1.2 – Периферійний пристрій відстеження м’язової активності

Nintendo Wii Remote – це контролер, який може визначати свої власні переміщення в тримірному просторі. Він містить динамік та вібруючий механізм, що надає додатковий зворотній зв’язок (Рис. 1.3). Детектор руху дозволяє користувачам керувати процесами за рахунок рухів рук, обробку яких виконує акселерометр та світлочутлива матриця. Матриця сприймає інфрачервоне світло, яке передається з спеціальної панелі Wii Sensor Bar, що встановлюється вище або нижче екрану пристрою. Wii Remote має безпровідне підключення до консолі за допомогою Bluetooth, що надає свободу рухів. Сам контролер може комбінуватись з багатьма спеціально випущеними Wii аксесуарами, такими як: рулі, пістолети, ракетки та інші [5]. Перевагами цього рішення була його революційність, адже до появи Wii Remote ніхто не міг надати широкій публіці подібний ігровий досвід. З недоліків варто зазначити залежність від контролера, адже саме його рухи розпізнає камера, а тому у випадках коли гравець далеко від камери або контролер у нього за спиною – система працює некоректно або ж взагалі не працює.

Рисунок 1.3 – Периферійний пристрій відстеження рухів з використанням контролера для керування ігровою приставкою

PlayStation Move – чутливий до руху контролер для ігрових приставок серії PlayStation фірми Sony. Система працює наступним чином: в приміщенні кріпиться камера PlayStation Eye, яка відстежує рухи контролера в тривимірному просторі за світінням кульки на кінці контролера і розпізнає образи (Рис. 1.4). Перевага, за словами інженерів Sony, у тому що їх технологія дозволяє досягти феноменальної точності при відносній дешевизні матеріалів і процесу виготовлення. Недоліки ті ж, що і у Nintendo Wii – контролер, який міг порушити ігровий процес, зникнувши з поля зору камери [6].

Рисунок 1.4 – Периферійний пристрій відстеження рухів з використанням камери та контролерів для керування ігровою приставкою

Microsoft Kinect – безконтактний сенсорний ігровий контролер, розроблений для ігрових приставок серії Xbox, а згодом адаптований до використання на персональних комп’ютерах. Представляє собою поєднання апаратного і програмного рішень у вигляді горизонтальної коробки розміром 23х7 см, що розміщується вище чи нижче екрана. Дозволяє користувачеві взаємодіяти з приставкою або комп’ютером через пози тіла, виконувані людиною фігури і усні команди (Рис. 1.5). Перевагами технології є система сенсорів глибини, що дозволяє не використовувати нічого зайвого в процесі гри – достатньо лише рухів людини. Великим недоліком стала потреба розробляти ігри спеціально для Kinect, тобто не було можливості адаптувати будь-які уже існуючі ігри до цієї системи [7].

Рисунок 1.5 – Периферійний пристрій відстеження рухів з використанням камери та інфрачервоного датчика для керування у ком’ютерних іграх

Розробка Microsoft зацікавила багатьох користувачів Xbox, однак продажі уже починаючи з першого року після виходу почали стрімко знижатись. Причин на те було декілька. Перша – мала бібліотека ігор, сумісних з технологією Kinect. Друга причина посилила першу – більшість з цих ігор вийшли нецікавими і одноманітними. Зазвичай це були прості активності, ігровий процес яких зводився до постійного наведення рук на різні визначені області екрану. Таких розваг в кращому випадку вистачало на 10-15 хвилин, потім ставало нудно. Третя ж причина стала останньою краплею для гравців. Справа у тому, що розпізнавання рухів працювало неналежним чином в різних іграх. І це залежало не від камери Kinect, а від самих ігор, адже оптимізація роботи системи були під відповідальністю розробників цих ігор. Все це спричинило поступову втрату інтересу користувачів, і в 2017 році Microsoft заявили, що припиняють виробництво Kinect.

У 2013 дослідники з Китаю створили Kinect Language Translator – прототип системи, яка розуміє жести мови жестів і перетворює їх на усну та письмову мови – і навпаки. Система фіксує розмову з обох сторін: письмовий та усний переклад робиться в режимі реального часу, тоді як система бере вимовлені слова слухаючого і перетворює їх у точні, зрозумілі знаки. Проект став результатом

співпраці між Китайською академією наук, Пекінським університетом союзу та Microsoft Research Asia, кожна з яких зробила вирішальний внесок. З цією технологією комунікація між глухими та людьми зі здоровим слухом стала природнішою, ніж з використанням перекладачів або письмового спілкування. Людина з вадами може самостійно висловлювати думку, наповнюючи її власними переживаннями та емоціями [8].

Kai: Gesture Controller – пристрій для керування на основі жестів ноутбуками, VR, дронами, смарт-телевізорами чи мобільними пристроями (Рис. 1.6). Kai можна носити на долоні, він довзоляє здійснювати цифрові взаємодії за допомогою жестів. Kai відстежує рухи руки та кожного окремого пальця, щоб розпізнавати та активувати безліч жестів. Простий, функціональний і недорогий пристрій, який допомагає в будених справах, чутливий і з довготривалим життям батареї [9].

Рисунок 1.6 – Периферійний пристрій розпізнавання жестів з відстеженням окремих пальців

Такий контролер рухів постачається з вбудованими 10 жестами, які можна видозмінювати (Рис. 1.7). Ці рухи допомагають під час керування презентаціями, використання VR пристроїв, користуванням планшетами, телевізорами чи телефонами.

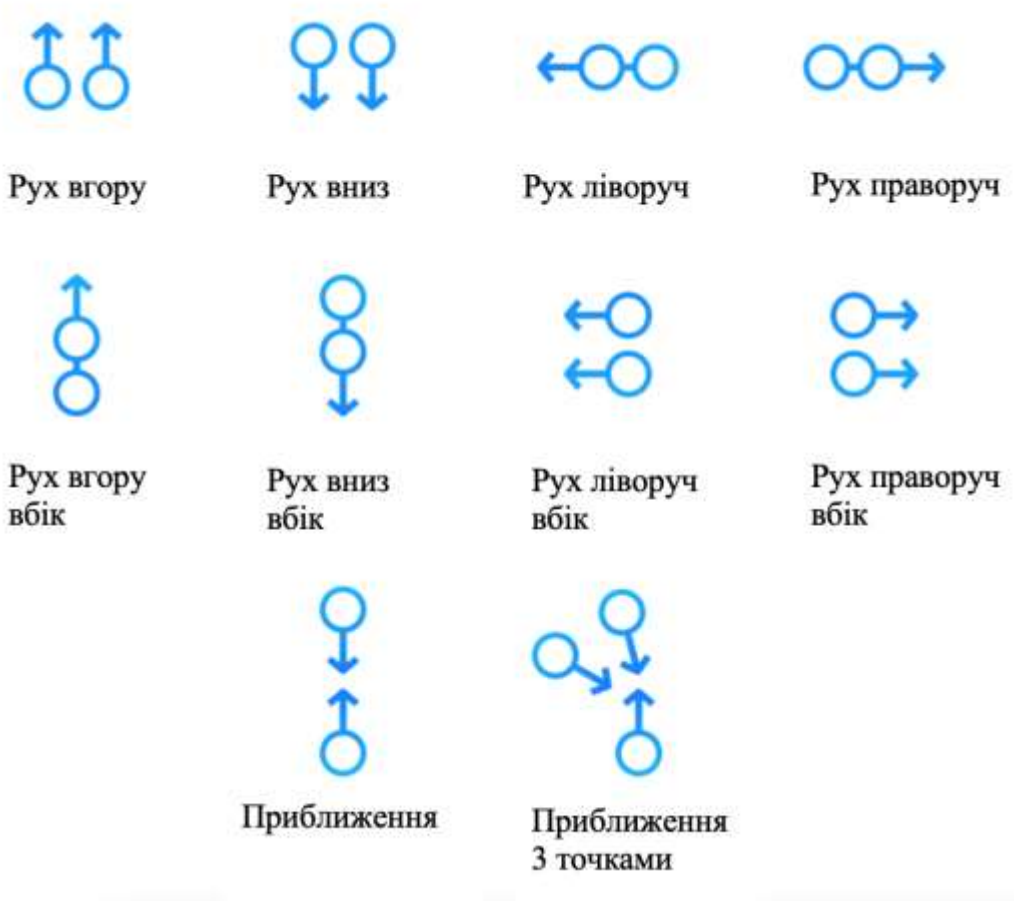


Рисунок 1.7 – Жести периферійного пристрою розпізнавання жестів з відстеженням окремих пальців

Oculus Touch — це пара гусеничних контролерів, які створюють ефект «присутність руки» — відчуття, що віртуальні руки насправді належать користувачу (Рис. 1.8). Констелляційне відстеження дає змогу керувати об’єктами у вашому віртуальному середовищі з надзвичайною точністю. Це виводить взаємодію на новий рівень. Дотик забезпечує природну присутність руки, що вимагає небагато думок про те, як використовувати віртуальні руки чи що з ними

робити. Дотик дозволяє точніше керувати будь-якими об’єктами у віртуальному просторі. Вловлюючи нюанси людського виразу, Touch обробляє жести, як-от вказувати, махати рукою та піднімати великий палець. Сенсорні контролери забезпечені традиційними кнопками дій, джойстиками та аналоговими тригерами, які створюють новий досвід. Це елементи керування для хапання, підйому тощо [10].

Рисунок 1.8 – Периферійний пристрій для керування у комп’ютерних іграх за допомогою рухів та кнопок

Тар Strap 2 — це універсальний пристрій для керування клавіатурою, мишкою та жестами, який можна носити однією рукою. Він дозволяє керувати своїми пристроями протягом 10 годин після повної зарядки (Рис. 1.9). Має 5 режимів: режим клавіатури – вводьте літери, цифри, символи та символи на своїх смарт-пристроях, на будь-яких поверхнях; режим AirMouse – введення та керування за допомогою жестів у повітрі на будь-якому пристрої Bluetooth; режим оптичної миші – точна оптична миша з роздільною здатністю 1000 DPI забезпечує навігацію, вибір, прокручування, перетягування та опускання в будь-якому середовищі на будь-якій поверхні; режим контролера – перетворює складні команди на прості дотики пальцями та рухи в повітрі, для керування програмами,

іграми та пристроями; спеціальний режим – можна налаштувати повністю на свій лад [11]. Використання безкоштового, простого і швидкого веб-інструменту TapMapper, SDK з відкритим вихідним кодом, API або плагіну Unity, для адаптації свої Taps до власних потреб.

Рисунок 1.9 – Периферійний пристрій визначення відстані та напрямку руки

Motion Sensor Kit – простий прилад, який може використовуватись для кодування, створення і користування у різних галузях (Рис. 1.10). За його допомогою можна створювати музику та ігри, керуючи лише рухами рукою. Набір датчиків руху вимірює відстань і напрямок. Він знає, наскільки близько ваша рука (або інші предмети), і перетворює ці дані. Датчик підключається до Mac, PC, Kano Computer Kit і Pixel Kit. Окрім використання Motion Sensor Kit для управління програмами, можна створювати свої задачі, використовуючи блокове кодування на основі подій, яке є чудовим вступом до реального коду з простим використанням блоків. Також є можливість перегляду коду, який створено в JavaScript [12].

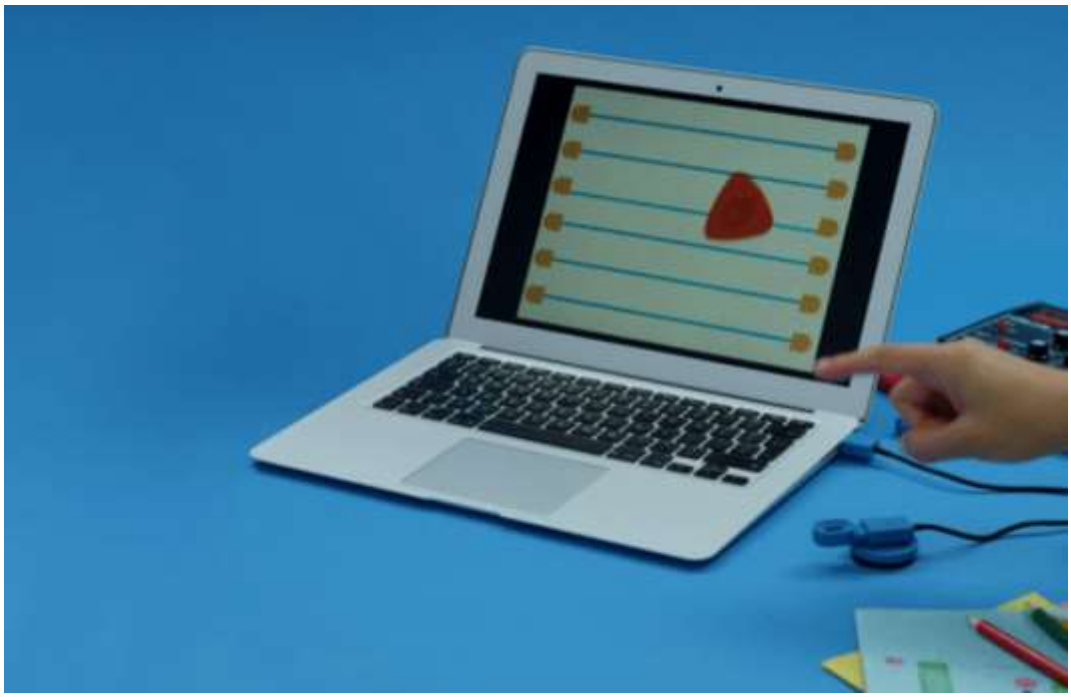


Рисунок 1.10 – Периферійний пристрій з можливістю гнучкого застосування технології розпізнавання жестів

HGR using Deep Learning in MatLab – це проєкт створений розпізнавати жести з використанням методів глибокого навчання. Використовується база даних ASL, яка містить жести рук. Також зображення попередньо обробляється для виділення жесту в статичному зображені. Цей пристрій заснований на мережі ConvNet, яка в свою чергу користується значною популярністю для глибокого навчання у виконанні завдань класифікації фото, відео, тексту та звуку (Рис.1.11). Згорткові нейромережі дають чудові результати для визначення шаблонів на зображенні, що дозволяє розпізнавати жести рук, обличчя та будь-який предмет [13]. Перевагою цієї моделі є те, що вона не потребує жодних алгоритмів виділення ознак. Однак, це робить її інваріативною до обертання розпізнаваного об’єкта (руки).

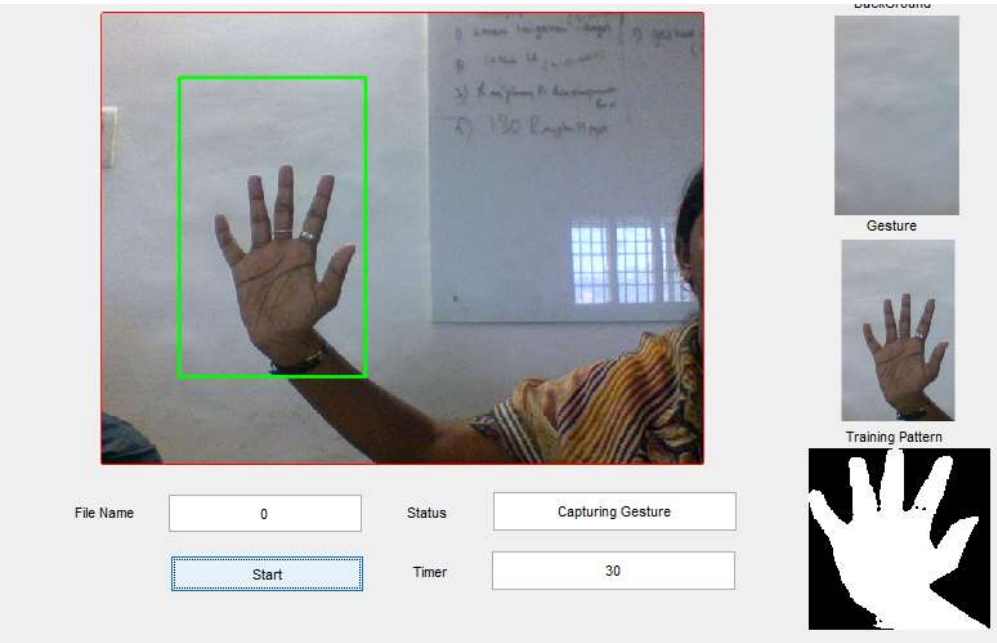


Рисунок 1.11 – Програмне забезпечення розпізнавання жестів з використанням згорткової нейронної мережі

1.3 Постановка задачі досліження

Визначимо задачі і вимоги до інтелектуальної системи розпізнавання жестів для керування комп’ютерними іграми. Задача: розширення функціоналу технології розпізнавання жестів.

Інформаційна технологія має реалізовувати такі функції:

- отримання на вході відеопотоку з веб-камери;
- аналіз вхідного потоку;
- класифікацію отриманих жестів;
- інтерпретацію жестового руху у команду керування комп’ютерною грою;
- виконання команди керування.

Також необхідно провести тестування розробленої системи на наявність помилок, безпечність та коректність роботи. Усі наявні елементи повинні чітко працювати без будь-яких помилок.

Початком розробки інтелектуальної системи розпізнавання жестів для керування комп’ютерними іграми є вибір методів розпізнавання образів та моделей

для навчання нейронних мереж. Після цього буде розроблено загальної структурної схеми системи, розробка загальної схеми алгоритму роботи системи, створення UML-діаграм для деталізації складових подальшої розробки. Наступним етапом є вибір технологій реалізації та безпосередня розробка інтелектуальної системи. Заключними етапами є тестування розробленої інтелектуальної системи і розробка інструкції користувача.

1.4 Висновок до розділу 1

Обгрунтовано актуальність проблеми розпізнавання жестів для задачі керування комп’ютерними іграми.

Виконано огляд сучасних систем розпізнавання жестів та здійснено аналіз їх основних переваг та недоліків. Аналіз показав, що абсолютна більшість комерційно успішних продуктів потребують додаткових аксесуарів для отримання жестів людини та інтерпретації їх у команди керування грою. Потреба у додатковому оснащенні з датчиками викликана здебільшого проблемою недостатньої точності розпізнавання в умовах використання лише камери. Тому є актуальним переосмислення застосування технології жестового керування у комп’ютерних іграх та робота над підвищенням точності розпізнавання.

Визначено задачі та вимоги до інформаційної технології розпізнавання жестів для керування у комп’ютерних іграх.

2 ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ КЕРУВАННЯ КОМП'ЮТЕРНИМИ ІГРАМИ

2.1 Варіантний аналіз шляхів розв'язання поставленої задачі

Розпізнавання образів – науковий напрямок, пов'язаний з розробкою принципів і побудовою систем, призначених для визначення належності даного об'єкту до одного із заздалегідь виділених класів. Іншими словами, це процес віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані із загальної маси несуттєвих даних. Традиційно завдання розпізнавання образів класифікують як задачу штучного інтелекту, зокрема машинного навчання. Навчання – процес вироблення в певній системі потрібної реакції на групи зовнішніх ідентичних сигналів шляхом багаторазового впливу на систему зовнішнім коректуванням. Такий вплив у навчанні прийнято називати «заохоченнями» та «покараннями». Спосіб генерації цього коректування визначає безпосередньо алгоритм навчання. Самонавчання відрізняється від навчання тим, що при ньому додаткова інформація про вірність реакції системі не повідомляється [14].

Згорткова нейронна мережа складається з шарів вузлів, що містять вхідний шар, один або кілька прихованих шарів та вихідний шар. Кожен вузол підключається до іншого і має відповідну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані далі не передаються. Згорткові нейронні мережі характеризуються своєю чудовою продуктивністю із входами зображення, мови або звукового сигналу. Вони добре розв'язують завдання комп'ютерного зору. Комп'ютерний зір – це область штучного інтелекту, яке дозволяє комп'ютерам та системам отримувати значущу

інформацію з цифрових зображень, відео та інших візуальних входів, і на основі цих входів він може вживати заходів. Здатність надавати рекомендації відрізняє комп'ютерний зір від стандартних завдань розпізнавання зображень.

За досвідом науковців, що роками працюють з такими нейромережами, оптимальні результати у розпізнаванні образів демонструють мережі з нескладною архітектурою, що складаються з шару прихованих нейронів та нейронів-виходів. На рисунку 2.1 представлено схему згорткової нейромережі.

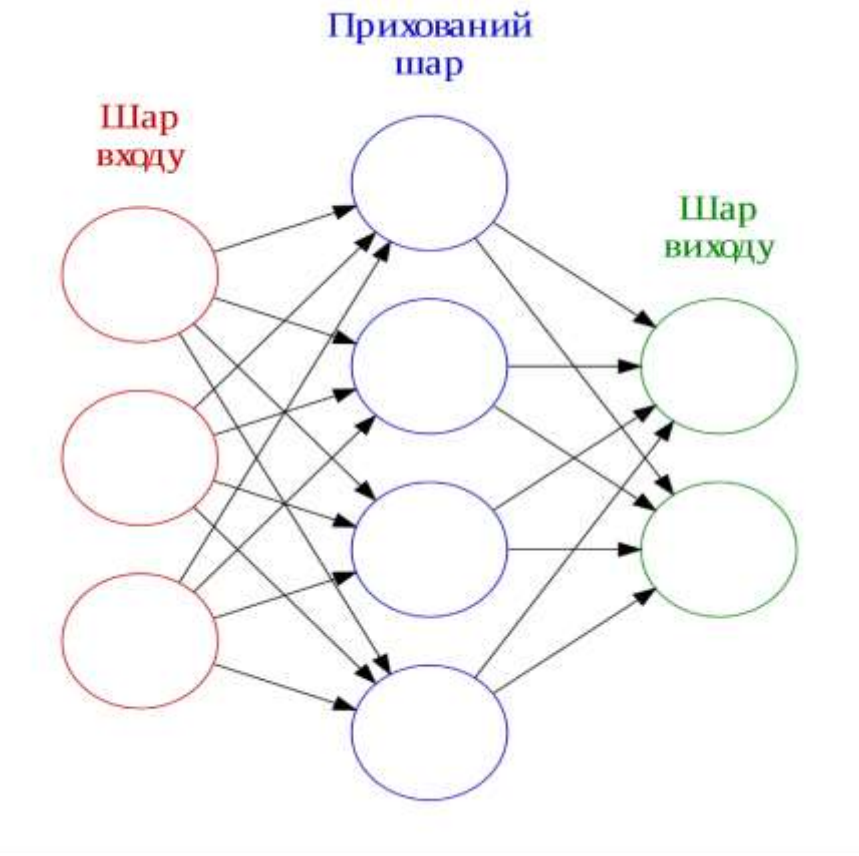


Рисунок 2.1 – Структурна схема згорткової мережі для розпізнавання жестів

Найбільш часто в задачах розпізнавання образів розглядаються монохромні зображення, що дає можливість розглядати зображення як функцію на площині. Якщо розглянути точкову множину на площині T, де функція $x(x, y)$ виражає в кожній точці зображення його характеристику – яскравість, прозорість, оптичну щільність, то така функція є формальним записом зображення. Множина всіх

можливих функцій $x(x, y)$ на площині T, є моделлю множини всіх зображень X. Уводячи поняття подібності між образами можна поставити задачу розпізнавання.

Образ – класифікаційне угруповання в системі класифікації, яка об'єднує (виділяє) певну групу об'єктів за певною ознакою. Образи мають характерну властивість, що виявляється в тому, що ознайомлення з кінцевим числом явищ з однієї і тієї ж множини дає можливість дізнаватися як завгодно велике число її представників. Образи мають характерні об'єктивні властивості в тому сенсі, що різні люди, які навчаються на різному матеріалі спостережень, здебільшого однаково і незалежно один від одного класифікують одні і ті ж об'єкти [15]. У класичній постановці задачі розпізнавання універсальна множина розбивається на частини-образи. Кожне відображення будь-якого об'єкта на сприймаючі органи системи, що розпізнає, незалежно від його положення щодо цих органів, прийнято називати зображенням об'єкта, а множини таких зображень, об'єднані певними загальними властивостями, являють собою образи. Методика віднесення елемента до якого-небудь образу називається вирішальним правилом. Ще одне важливе поняття – метрика, спосіб визначення відстані між елементами універсальної множини. Чим менший цей період, тим більше схожими є об'єкти (символи, звуки та ін.) – те, що ми розпізнаємо. Зазвичай елементи задаються у вигляді набору чисел, а метрика - у вигляді функції. Від вибору уявлення образів і реалізації метрики залежить ефективність програми, один алгоритм розпізнавання з різними метриками буде помилятися з різною частотою.

Першочерговим завданням у багатьох методах класифікації об'єктів у реальному часі є інтегральне представлення зображення. Інтегральне представлення дозволяє швидко розраховувати сумарну яскравість довільного прямокутника, причому час розрахунку не залежить від площі прямокутника. Інтегральне представлення зображення являє собою матрицю, розмірність якої збігається з розмірністю вихідного зображення. У кожному елементі матриці зберігається сума інтенсивностей всіх пікселів, що знаходяться лівіше і вище даного елемента.

Елементи матриці розраховуються за такою формулою:

$$L(x,y) = \sum_{i=0,j=0}^{i\le x,j\le y} * I(i,j),$$

де $I(i,j)$ — яскравість пікселя вхідного зображення.

Кожен елемент матриці $[x, y]$ являє собою суму пікселів в прямокутнику від (0,0) до (x, y), тобто значення кожного пікселя (x, y) дорівнює сумі значень усіх пікселів лівіше і вище даного пікселя (x, y). Розрахунок матриці займає лінійний час, пропорційне числу пікселів в зображенні, тому інтегральне зображення прораховується за один прохід. Розрахунок матриці можна виробляти за рекурентною формулою:

$$(x, y) = I(x, y) – L(x - 1, y - 1) + L(x, y - 1) + L(x - 1, y).$$

Для того, щоб впізнати образ і віднести його до певного виду, потрібно виконати класифікацію. Універсальним способом класифікації вхідних даних є форми. Для такої класифікації застосовують шаблони або аналітичний опис. Процес розпізнавання відбувається за наступною послідовністю: система аналізує функції вхідних форм-образів і зберігає результати як ключові моменти у профілі розпізнавання. Цей процес називається індексацією. Після цього порівнюються зображення в мережевому трафіку або збережені в сховищах даних із проіндексованими образами. Ступінь відповідності виявленої форми ключовим точкам в проіндексованій порожній формі називається вирівнюванням. У більшості систем 85% ключових точок повинні збігатися або вирівнюватися, щоб форма вважалася збігом. Найчастіше використовується середньоквадратична різниця значень яскравості зображень шаблону і аналізованого кадру:

$$diff = \frac{\sqrt{\sum_{i=0}^{\infty}(pixelI1_i - pixelI2_i) \times (pixelI1_i - pixelI2_i)}}{n-1}.$$

Велику кількість об'єктів можна класифікувати залежно від їх кольору: вони або постійно мають певне забарвлення, або в деякі моменти їх забарвлення може бути визначене достатньо чітко. У зв'язку з тим, що існує безліч колірних моделей (RGB, YUV, CMYK, HSV і т.д.), нерідкі випадки, коли в одній чи іншій моделі

даний об'єкт можна класифікувати майже безпомилково. Однак інформація про те, яку саме колірну модель використовувати і як краще організувати пошук об'єкта, маючи в розпорядженні зображення в даному базисі, часто може бути отримана лише експериментальним шляхом [16]. Щодо методів розпізнавання образів, можна виділити наступні:

– Метод перебору. Він ґрунтується на порівнянні вводу з базою даних, де для кожного класу об'єктів відображені різні модифікації представлення. Для оптичного розпізнавання образів можна застосувати метод перебору виду об'єкта під різними кутами, масштабами, зміщеннями, деформаціями. При розпізнаванні букв треба перебирати шрифт, його розмір, колір, кут нахилу і інші властивості. У випадках розпізнавання звукових образів, відповідно, відбувається порівняння з деякими уже відомими шаблонами.

– Глибокий аналіз характеристик. В задачах оптичного розпізнавання це можуть бути додаткові, більш детальні геометричні чи тонові характеристики. При роботі зі звуковими об’єктами проводять частотний аналіз, вивчення амплітуди тощо.

– Застосування моделі персептрона. Дану модель представив Френк Розенблат, ввівши в модель Маккаллока і Пітса властивість зв`язків до модифікації, зробивши її придатною до навчання. Спочатку персептрон представляв собою одношарову структуру з жорсткою пороговою функцією процесорного елемента та багатозначними входами. Складніші задачі потребували потужніших рішень, тому виходом із ситуації стали багатошарові персептрони. Багатошаровими персептрони називають нейронні мережі прямого поширення. Вхідний сигнал в таких мережах поширюється від шару до шару. Багатошаровий персептрон в загальному уявленні складається з наступних елементів: множини вхідних вузлів, які утворюють вхідний шар; одного або декількох прихованих шарів обчислювальних нейронів; одного вихідного шару нейронів. Персептрони не володіють здатністю до чистого узагальнення, але вони цілком задовільно функціонують в експериментах з розрізненням образів [17].

2.2 Обгрунтування вибору моделей навчання нейронних мереж

Машинне навчання – це процес, в результаті якого система поступово набуває здатність відповідати потрібними реакціями на певні сукупності зовнішніх впливів, а адаптація – це підстроювання параметрів і структури системи з метою досягнення необхідної якості управління в умовах безперервних змін зовнішніх умов.

У машинному навчанні виділяють дві основні техніки – навчання з учителем і навчання без учителя. Для навчання з учителем використовують вхідні і відповідні їм вихідні дані для тренування, завдяки чому мережа вчиться передбачати вихідні дані за вхідними. При навчанні без учителя модуль отримує лише вхідні дані, навчаючись знаходити в них явні і приховані зв’язки. У свою чергу, навчання з учителем поділяється на класифікацію і регресію. Мережу навчають класифікації у випадках, коли потрібно передбачити, до якої із відомих уже категорій належить об’єкт із вхідних даних. Регресія ж потрібна для визначення неперервних величин. Регресивні методи навчання частіше за все використовують для безпосереднього прогнозування, наприклад, погоди або курсу валют. Для виконання задач розпізнавання образів однозначно найбільше підходить метод класифікації.

Перейдемо до аналізу моделей навчання. Почати варто з байєсового класифікатора, адже він є оптимальним з точки зору статистики. В ідеальній реалізації, результат його класифікації неможливо покращити: у випадках, коли можлива однозначна відповідь – він її дасть. Коли ж відповідь неоднозначна – результат характеризує кількісна міра цієї неоднозначності. Однак, для такої ідеальної реалізації потрібна вибірка з усіма можливими комбінаціями змінних. Розмір такої вибірки зі збільшенням кількості змінних зростатиме за експонентою і потребуватиме величезних обчислювальних потужностей. Проблему вирішує наївний байєсів класифікатор, який ґрунтується на припущенні про незалежність змінних. Він дозволяє не вивчати всі можливі зв’язки між ними, обмежуючись

лише впливом кожної змінної окремо. У теорії – модель добре працює тільки коли припущення спрацьовує і змінні дійсно виявляються незалежними. На практиці ж результат її роботи продовжує корелювати навіть при вагомій залежності між змінними. У таблиці 2.1 наведено характеристику байєсового навчання.

Таблиця 2.1 - Характеристика моделі байєсовського навчання

Переваги	Недоліки
Робота з малим обсягом даних	Передбачає обмежену оптимізацію
Гнучкість	Часто важко задати адекватну апріорну величину

Генеративні моделі базуються на алгоритмах, які прагнуть до цілісного моделювання процесу без відкидання будь-якої інформації. Це означає, що мережа не просто відносить об’єкти до того чи іншого класу. Вона вчиться розуміти що ці класи собою являють. Навідміну від генеративних моделей, дискримінативні не заглиблюються в суть класів, однак часто вузькоспеціалізовані дискримінативні моделі отримують набагато кращі результати, ніж узагальнені генеративні.

На рисунку 2.2 наведено принцип роботи генеративної та дискримінативної моделей.

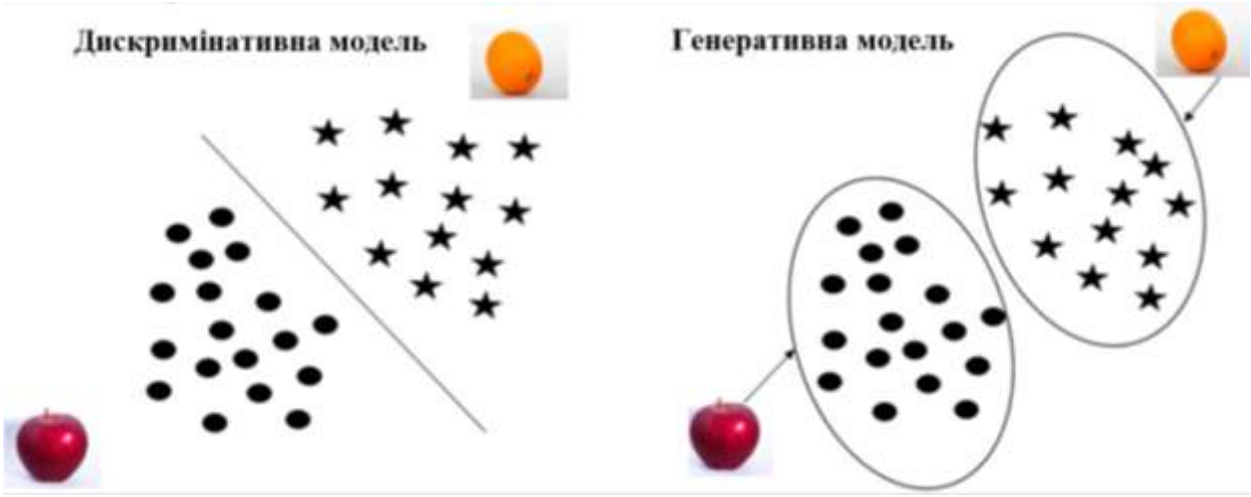


Рисунок 2.2 – Приницпова схема генеративної на дискримінативної моделей

Найпоширенішим прикладом навчання з використанням генеративних алгоритмів є генеративна змагальна мережа. Така мережа, в свою чергу, складається з двох нейронних мереж, які виконують протилежні задачі, при цьому змагаючись і створюючи умови розвитку одна для одної. Одна з мереж генерує зразки на основі тренувального набору. Задача іншої полягає у тому, щоб відрізнити правильні оригінальні зразки від згенерованих першою мережею. Таким чином, мережі допомагають одна одній навчатись. Успішним результатом буде момент, коли мережа-генератор навчиться створювати правильні зразки [18]. У таблиці 2.2 наведено характеристику генеративної моделі.

Таблиця 2.2 - Характеристика генеративної моделі

Переваги	Недоліки
Має меншу обчислювальну складність ніж байєсова модель	Часто не дозволяє усунути проблеми байєсівського підходу
Мова графа залишається природньою, що допомагає отримувати апріорну величину	Результати швидко погіршуються у зв’язку із систематичною помилкою в трактуванні умовної незалежності

Завданням алгоритмів моделі навчання на базі ядер є пошук і вивчення загальних типів відношень у наборах даних. Це можуть бути кластери, рейтинги, класифікації, кореляції. Для багатьох алгоритмів дані у необробленому вигляді мають бути перетворені у подання векторів ознак. Будь-яку лінійну модель можна перетворити на нелінійну модель, замінивши її функції функцією ядра. Навчання відбувається на базі екземплярів. Замість того, щоб вивчати певний сталий набір параметрів, що відповідає особливостям їх вхідних даних, ядра просто запам'ятовують приклад навчання та вивчають для нього відповідний ваговий прогноз. У таблиці 2.3 наведено характеристику моделі навчання на базі ядер [19].

Таблиця 2.3 - Характеристика моделі навчання на базі ядер

Переваги	Недоліки
Алгоритми навчання є досить практичними при певних обчислювальних складностях	Не забезпечує достатню ефективність при великих обсягах даних

Побудова дерева рішень виконує завдання класифікації, регресії і опису об’єктів в різних сферах, пов’язаних з обробкою даних. У деревах рішень чітко прослідковуються причинно-наслідкові зв’язки. Метою навчання на основі дерева рішень є створення моделі, яка передбачає значення цільової змінної, ґрунтуючись на деяких вхідних змінних. Кожен із внутрішніх вузлів відповідає одній з вхідних змінних. Є ребра до нащадків для кожного можливого значення цієї вхідної змінної. Кожен лист є значенням цільової змінної, яке визначається значеннями вхідних змінних від кореня до листа [20].

На рисунку 2.3 – Принцип дерева рішень показано приклад дерева рішень задачі виживання на кораблі «Титанік». Першою умовою є стать, якщо не чоловіча – то вірогідність виживання. Якщо чоловіча – тоді важливою є наступна умова, чи досягнуть людиною віку більше 9,5 років. Якщо відповідь негативна, то вірогідність смерті, якщо стверджувальна, тоді виникає наступне питання. При кількості родичів на борту більше 2,5 вірогідність виживання, якщо менше, то смерть.

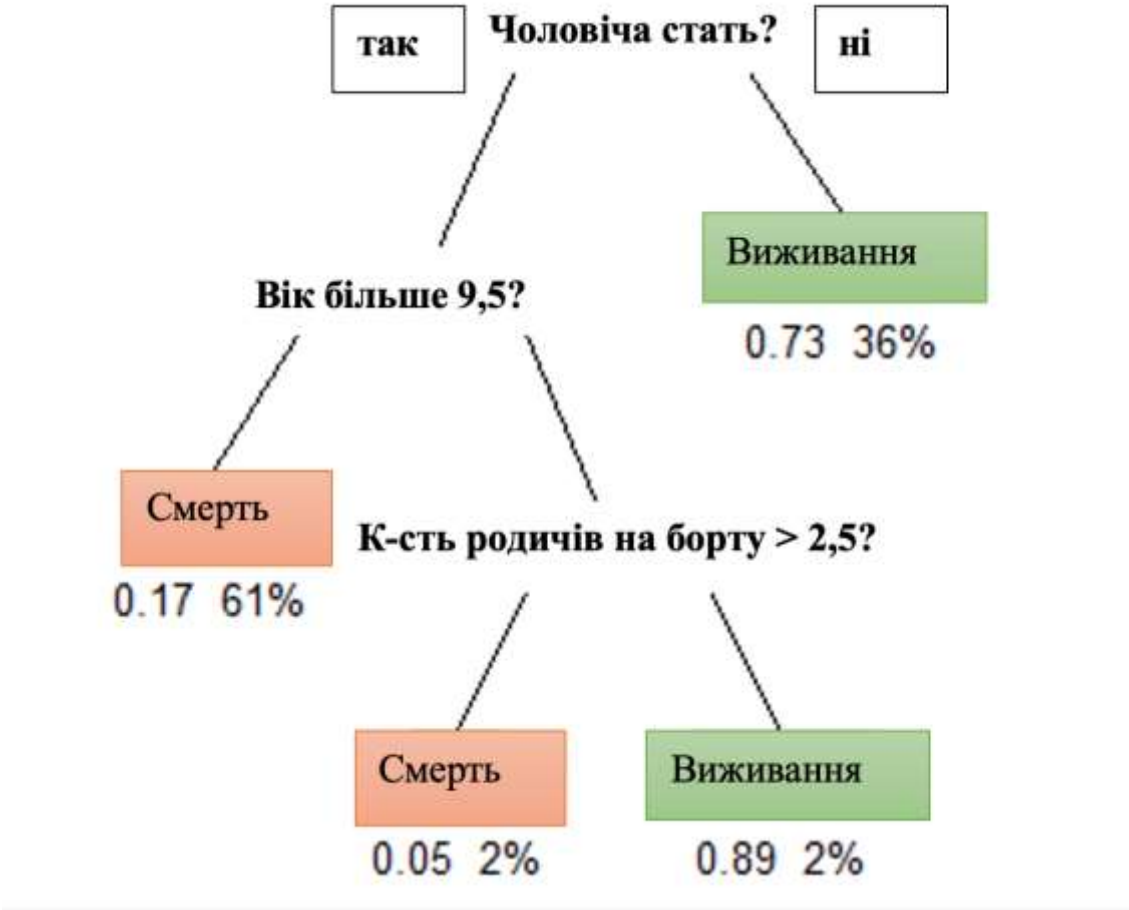


Рисунок 2.3 – Принцип дерева рішень

У таблиці 2.4 наведено характеристику дерева рішень.

Таблиця 2.4 – Характеристика моделі на основі дерева рішень

Переваги	Недоліки
Високі рівень автоматизації і швидкість роботи	Теоретичний опис для багатьох алгоритмів відсутній
	Через ієрархію можуть програвати в продуктивності іншим моделям

Градієнтний спуск – один із найпопулярніших у використанні алгоритм навчання. Це ітераційний алгоритм, на кожному кроці якого вектор ваг змінюється

у напрямку найбільшого убування цільового функціоналу, тобто в напрямку антіградієнта, Базується на знаходженні мінімального значення функції втрат. Мінімізація будь-якої функції означає пошук найглибшої точки на графіку цієї функції. Пошук мінімуму означає отримання найменшої можливої похибки або підвищення точності моделі. Точність збільшується за рахунок перебору навчальних даних при налаштуванні ваг і зміщень [21].

На рисунку 2.4 наведено діграму оптимізуючої функції градієнтного спуску.

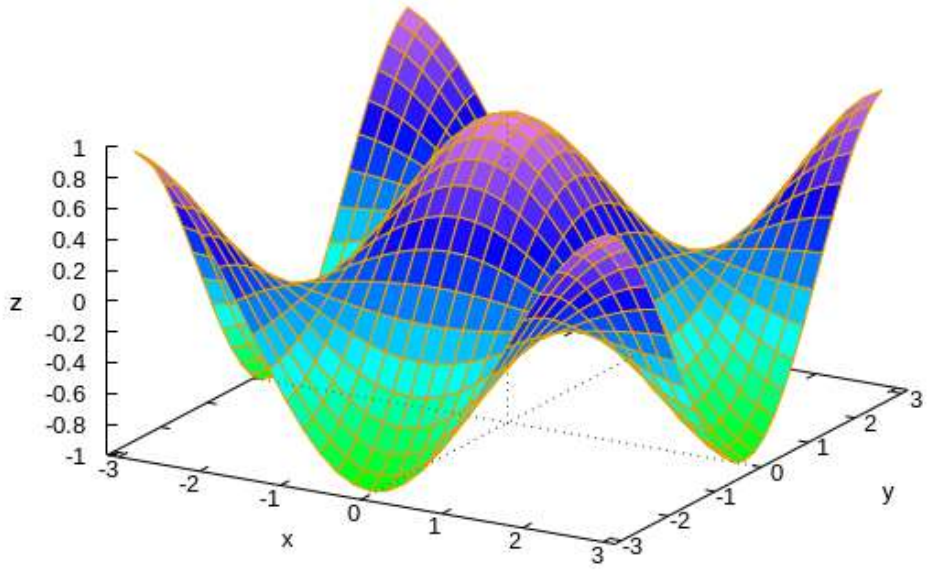


Рисунок 2.4 – Діграма оптимізуючої функції градієнтного спуску

У таблиці 2.5 наведено характеристику навчання за градієнтним спуском.

Таблиця 2.5 – Характеристика моделі навчання за градієнтним спуском

Переваги	Недоліки
Невисока обчислювальна складність	Низька здатність до перенавчання
Модульність	Незадовільна документація

Навчання з підкріпленням – це унікальний приклад моделі, яку можна трактувати і як навчання з учителем, і як навчання без нього. Справа у тому, що в ролі учителя виступає не людина, а середовище, в якому вона опиняється. Мережа взаємодіє з середовищем і отримує від нього зворотній зв’язок. Таким чином вона навчається виживати. Перша ціль в навчанні з підкріпленням - мінімізувати помилки.

Мережа вчиться аналізувати інформацію перед кожним наступним ходом. Друга мета - отримати максимальну вигоду від виконання поставленої задачі. Сама ж вигода при цьому повинна бути визначена і задана системі заздалегідь: наприклад, максимально швидкий час проходження маршруту або оптимальне використання наданих ресурсів.

У таблиці 2.6 наведено характеристику навчання з підкріпленням.

Таблиця 2.6 – Характеристика моделі навчання з підкріпленням

Переваги	Недоліки
Відмінно проявляє себе в невеликих моделях світу	Тривала оцінка кількості станів
Хороший варіант для застосування в області комп’ютерних ігор	Всі відомі алгоритми погано масштабуються за кількістю прихованих станів

Із методів навчання обрано одну із варіацій градієнтного спуску, а саме метод оберненого поширення помилки. Він чудово працює зі згортковими нейронними мережами і найчастіше застосовується для багатошарового перцептрона. Головна ідея полягає в розповсюдженні сигналів помилки в оберненому напрямку руху звичайних сигналів, тобто від виходів мережі до її входів. Складнощі може доставити процес навчання, час якого неможливо визначити, і чим складніша система, тим довше це буде продовжуватись. Для розпізнавання кількох рухів руки

не знадобляться великі потужності, тому даний метод чудово підходить для використання у процесі виконання поставленої задачі.

2.3 Аналіз методів попередньої обробки зображення

Щоб збільшити точність знаходження контуру перед виділенням контурів, зображення потрібно піддати попередній обробці. При попередній обробці залежно від того, який метод виділення контурів буде використовуватися, можуть застосовуватися такі методи попередньої обробки зображення:

- згладжування, фільтрація шуму, підвищення контрасту;
- переведення зображення в напівтонове;
- видалення фону;
- бінаризація зображення;
- сегментації зображення.

Згладжування – це розмиття зображення для усунення шуму. Один з найважливіших етапів при виділенні контурів, так як більшість методів при високому рівні шуму на зображень виявляються непрацездатними і призводять до появи хибних контурів, чорних плям і розривів як при бінаризації зображення так і на інших етапах [22].

Переклад зображення у напівтонове. Так як при розпізнаванні жесту колір не несе значної інформації і кольорове зображення не буде використовуватися при аналізі і при порівнянні образів не використовуватимуться колірні критерії, а також у зв'язку з тим, що кольорове зображення важко сегментувати та аналізувати, то для подальшого аналізу зображення необхідно перевести в напівтонове, а потім здійснити бінаризацію.

При переведенні зображення з кольорового в напівтонове сіре використовується схема RGB to YUV. Для цього буде отримано інтенсивність (яскравість) за формулою:

$$Y = 0{,}299 \times R + 0{,}587 \times G + 0{,}114 \times B,$$

де *R*, G, B – червоний, зелений та синій канали відповідно.

Дана формула використовує коефіцієнти чутливості людського ока до кольорів RGB і дозволяє обчислити монохромну яскравість кожного окремого пікселя [23].

Так як працювати з напівтоновим зображенням не набагато легше, ніж на кольоровому, тому що воно містить багато зайвої інформації, то наступним кроком необхідно напівтонове зображення перевести в бінарне.

Бінаризація. Це операція порогового розподілу, результатом якої є намальоване зображення. Бінаризація потрібна для зменшення кількості інформації, що міститься на зображенні. Зазвичай вихідним зображенням при бінаризації є напівтонове зображення з деякою кількістю рівнів яскравості. Підсумком бінаризації є чорно-біле зображення, пікселі якого мають тільки значення 0 і 1. Бінарізація зазвичай відбувається за певним порогом і може відбуватися різними способами та з різними пороговими значеннями [24].

Виділяють такі види методів бінаризації:

Методи обробки з постійним порогом використовують постійний поріг для всього зображення (а деякі і взагалі незалежні від зображення порогові значення). Такі методи також називають глобальними або з глобальним граничним значенням. Серед методів із пороговим значенням виділяють:

Бінаризація з нижнім порогом – найпростіший метод порогової бінаризації з одним глобальним граничним значенням:

$$f' = \begin{cases} 0, f(m,n) \geq t; \\ 1, f(m,n) < t. \end{cases}$$

Суть даного методу полягає в тому, що значення менше порогового стають рівні одиниці (тобто білим кольором), а значення вище порога нулю (тобто чорним кольором). Мінусом цього методу є складність у знаходженні цього порогового значення.

Бінаризація з верхнім порогом – аналогічно бінаризації з нижнім порогом, але результатом є негатив зображення, отриманого в процесі бінаризації:

$$f' = \begin{cases} 0, f(m,n) \leq t; \\ 1, f(m,n) > t. \end{cases}$$

Бінарізація з подвійним обмеженням – даний метод використовується для виділення областей, в яких у потрібних пікселів значення яскравості змінюється в заданому діапазоні:

$$f' = \begin{cases} 0, f(m,n) \geq t_1; \\ 0, t_1 < f(m,n) \leq t_2; \\ 0, f(m,n) > t_2; \end{cases}$$

або

$$f' = \begin{cases} 1, f(m,n) \geq t_1; \\ 0, t_1 < f(m,n) \leq t_2; \\ 1, f(m,n) > t_2. \end{cases}$$

Неповна порогова обробка – цей метод використовується для відокремлення фону від основного зображення, залишаючи при цьому всі деталі, що спрощує подальший аналіз.

$$f'(m,n) = \begin{cases} f(n,m), f(m,n) > t; \\ 0, f(m,n) \leq t. \end{cases}$$

Багаторівневе граничне перетворення не є бінаризацією, адже в результаті його роботи ми отримуємо зображення, що не є бінарним, а складається з сегментів з різною яскравістю:

$$f'(m,n) = \begin{cases} 1, f(m,n) \in D_1; \\ 2, f(m,n) \in D_2; \\ \dots\dots \\ n, f(m,n) \in D_n. \end{cases}$$

Каскад Хаара – це набір ознак, для яких вважається їх згортка з зображенням. Ознака Хаара – відображення f: X => Df , де Df — множина допустимих значень ознаки. Якщо задані ознаки f_1, … , f_n, то вектор ознак x = (f_1 (x), … , f_n (x)) називається ознаковим описом об'єкта $x \in X$. Ознакові описи допустимо ототожнювати з самими об'єктами. При цьому множину $X = D f_1 * \dots * D f_n$ називають ознаковим простором. Обчислюваним значенням такої ознаки буде: $F = X - Y$, де X – сума значень яскравості точок закриваються світлою частиною ознаки, а Y – сума значень яскравості точок закриваються темної частиною ознаки [25]. Для їх обчислення доцільно застосувати поняття інтегрального зображення, розглянуте у розділі 2.1.

Метод Віоли-Джонса з використанням ознак Хаара є одним з кращих алгоритмів для вирішення задач виявлення об'єктів на зображеннях в реальному часі, використовуючи двомірні зображення. Особливістю ознак Хаара, є найбільша, в порівнянні з іншими ознаками, швидкодія. У стандартному методі Віоли-Джонса використовуються прямокутні ознаки, зображені на рисунку 2.4, вони називаються примітивами Хаара.

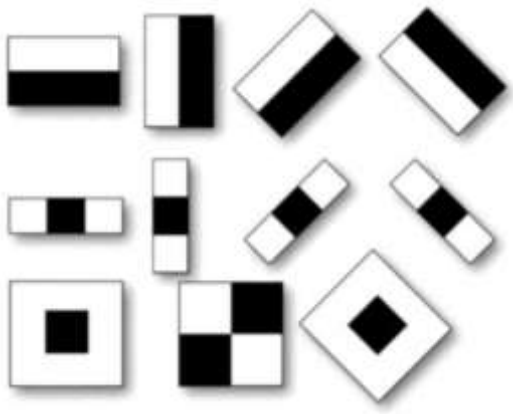


Рисунок 2.4 – Схема представлення примітивів Хаара

У розширеному методі Віоли-Джонса, що використовується в бібліотеці для задач комп’ютерного зору OpenCV, використовуються додаткові ознаки, зображені на рисунку 2.5.

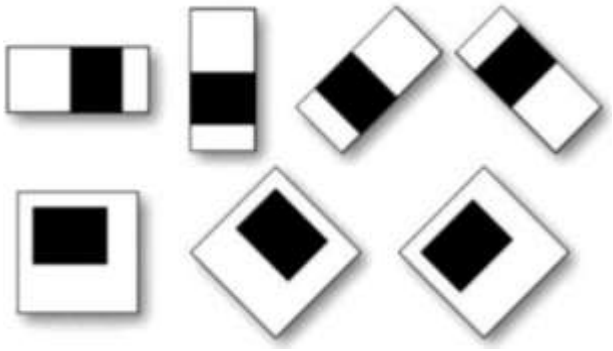


Рисунок 2.5 – Схема представлення додаткових ознак

Ознаки, запропоновані Віолою і Джонсом, містять більше однієї прямокутної області і дещо складніші за ознаки Хаара. Величина кожної ознаки обчислюється як сума пікселів у білих прямокутниках, з якої віднімається сума пікселів у чорних областях. Прямокутні ознаки більш примітивні, ніж керований фільтр, і, незважаючи на те, що вони чутливі до вертикальних і горизонтальних особливостей зображень, результат їх пошуку більш грубий. Однак при зберіганні зображення в інтегральному форматі перевірка прямокутної ознаки на конкретній позиції проводиться за константний час, що є їх перевагою порівняно з більш точними варіантами. Кожна прямокутна область у ознаках завжди суміжна з іншим прямокутником, тому розрахунок ознаки з 2 прямокутниками складається з 6 звернень в інтегральний масив, для ознаки з 3 прямокутниками – з 8, з 4 прямокутниками - з 9 [26].

Отже, процес вилучення ознак жестів відбувається наступним чином:

– перетворення зображення області жесту в напівтонове за допомогою інтегральної бінаризації;
– застосування до отриманого зображення розпізнавання на основі каскадів Хаара для визначення контурів руки;

– зміна розміру області жесту до 64 × 64 пікселів;

– Класифікація жеста.

На рисунку 2.4 наведено приклад обробки зображення за допомогою інтегрального представлення зображення та контуризації за Хаар-подібними характеристиками.

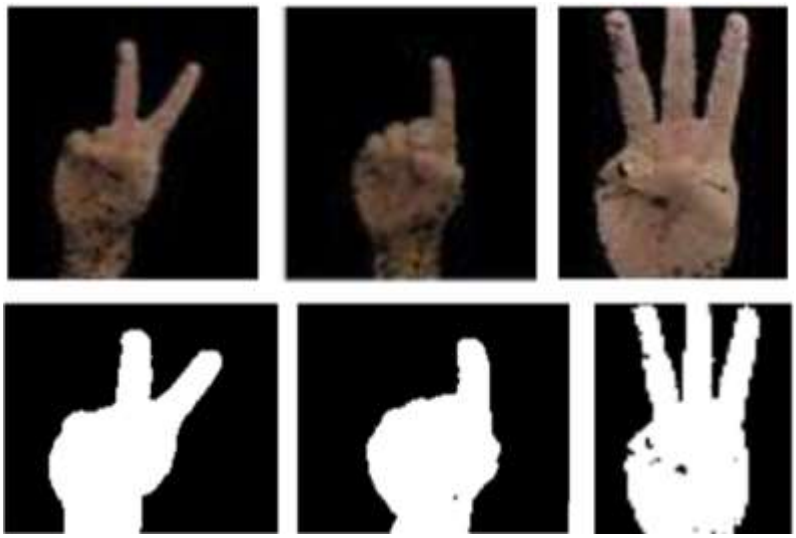


Рисунок 2.4 – Схема представлення бінаризованих зображень

За результатами досліджень розроблено метод виявлення жестів на півтонових зображеннях, який базується на комбінованому каскаді нейромережних класифікаторів (ККНК).

Для рівня ККНК, що призначений для верифікації жестів, застосовано згорткову нейронну мережу, яка в задачах класифікації володіє більшою стійкістю до деформацій вхідних образів, ніж інші відомі класифікатори, що дозволяє отримати високу достовірність виявлення. Для спрощення реалізації каскаду слабких класифікаторів запропоновано використати нейромережеву технологію, коли каскад класифікаторів являє собою багатошаровий персептрон, що складається із рівнів (нейронних шарів), кожний з яких містить один і більше слабких класифікаторів (нейронів). Входом для слабкого класифікатора є Хаарподібна ознака прямокутної форми, яка складається зі «світлих» і «темних» прямокутників, а її значення Feat розраховується за формулою:

$$Feat(x) = s_w \times SUM_w + s_b \times SUM_b,$$

де x – вхідне зображення, s_w та s_b – синаптичні ваги для всього прямокутника ознаки і для його темної частини відповідно, SUM_w та SUM_b – суми пікселів всього прямокутника ознаки і його темної частини відповідно. Вихідне значення слабкого класифікатора h знаходиться наступним чином [6]:

$$h(x) = \begin{cases} 1, & \text{якщо } Feat(x) < \theta; \\ -1, & \text{якщо } Feat(x) > \theta. \end{cases}$$

де θ – порогове значення слабкого класифікатора. У свою чергу вихідне значення КСК H являє собою лінійну комбінацію слабких класифікаторів [6]:

$$H(x) = \sum_{t=1}^{T} \eta_t \times h_t(x),$$

де T – кількість слабких класифікаторів, η_t – вага конкретного слабкого класифікатора.

2.4 Розробка загальної структурної схеми функціонування інформаційної технології розпізнавання жестів для керування у комп'ютерних іграх

Розробимо загальну структурну схему функціонування інформаційної технології розпізнавання жестів для керування комп'ютерними іграми. Вона складається з п'яти основних етапів:

- отримання вхідного зображення з веб-камери;
- бінаризація методом інтегрального представлення зображення;
- відстеження та виділення контурів руки за допомогою каскаду Хаара;
- класифікація жесту за допомогою згорткової нейромережі;
- інтерпретація розпізнаних жестів у команди керування грою.

Після бінаризації, на двотоновому зображенні за принципами ознак Хаара визначаються контури руки. Відстежується положення руки у межах екрану, адже її позиція впливає на команду керування, яку буде віддано грі.

Інтерпретатор команд отримує інформацію про класифіковані жести та співставляє їх з командами керування. На рисунку 3.1 зображено структуру інформаційної технології розпізнавання жестів для керування у комп’ютерних іграх.

Рисунок 3.1 – Загальна структурна схема функціонування інформаційної системи розпізнавання жестів для керування у комп’ютерних іграх

2.5 Висновок до розділу 2

Виконано аналіз можливих варіантів розв’язання задачі розпізнавання жестів. Доведено доцільність застосування згорткової нейромережі для розв’язання задачі розпізнавання жестів. Обґрунтовано доцільність застосування навчання нейромережі за градієнтним спуском, адже цей метод дозволяє мінімізувати обчислювальні потреби системи.

Проаналізовано методи попередньої обробки зображень. За результатами аналізу обрано каскад Хаара у поєднанні з інтегральним представленням зображення. Інтегральне перетворення дозволяє подати зображення у бінарному вигляді, при чому швидкість обробки залежить лише від розмірності зображення (а саме від кількості пікселів, яскравість яких треба обчислити). Для цього у процесі розпізнавання спочатку будуть визначатись контури руки, після чого область з розпізнаною рукою буде виокремлено із загального зображення у розмірі 64×64 пікселі. Це дозволить швидко та достовірно виконувати класифікацію жестів.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ КЕРУВАННЯ У КОМП'ЮТЕРНИХ ІГРАХ

3.1 Обґрунтування вибору інструментів технічної реалізації

Визначимо інструменти, за допомогою яких буде виконуватись програмна реалізація інформаційної технології розпізнавання жестів для керування у комп'ютерних іграх. Зокрема, необхідно визначити мову програмування для створення компонентів системи та віддати перевагу платформі для розробки.

Для порівняння було обрано 3 найпопулярніші мови програмування: Python, Java та C#. Наведемо особливості кожної з мов програмування та порівняльну характеристику цих мов у таблиці 3.1.

Java розроблялась як платформо-незалежна мова, тому вона має обмежену кількість низькорівневих можливостей для роботи з апаратним забезпеченням у порівнянні з іншими мовами. Наявний «Garbage collector», що допомагає запобігти витоку пам'яті шляхом видалення об'єктів, які більше не використовуються додатком. Підтримка узагальнень у даній мові реалізуються з використанням стирань. Параметри загального типу «стираються», а при компіляції в байт-код додаються приведення [27].

Python – це мова програмування загального призначення. Вона оптимізована для створення якісного програмного забезпечення, високої продуктивності праці розробників, перенесення програм і інтеграції компонентів. Мова Python використовується сотнями тисяч розробників по всьому світу в таких напрямках як створення веб-сценаріїв, системного адміністрування, створення користувацьких інтерфейсів, налаштування програмних продуктів під користувача, машинне навчання і багато іншого. Python – одна з найпопулярніших мов програмування в світі. Серед переваг, вона відрізняється простотою і легким синтаксисом, легко інтегрується з зовнішніми компонентами, написаними на інших мовах програмування, має мультипарадигменну архітектуру і підтримує об'єктно-

орієнтоване, функціональне та модульне програмування. А найголовніше – має велику кількість готових пакетів для аналізу даних та машинного навчання [28].

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML [29]. Однією з переваг мови C# є наявність великої кількості бібліотек та шаблонів, що значно економить час програміста. Як і у Java наявний «Garbage collector» та підтримка узагальнень. У цій мові є делегати, які слугують методами, що можуть бути викликані без повідомлення цільового об'єкта. Для досягнення такої ж функціональності в Java необхідно використовувати інтерфейс з одним методом або іншим способом обходу, який може вимагати великої кількості додаткового коду. У таблиці 3.1 наведено порівняльну характеристику мов програмування.

Таблиця 3.1 – Порівняльна характеристика мов програмування

Характеристика порівняння	Python	Java	C#
Швидкодія	+	-	+
Ручне управління пам’яттю	-	-	+
Перевантаження функцій	+	+	+
Наявність пакетів для машинного навчання	+	+	-
Шаблони	+	+	+

Для виконання задачі дипломної роботи було обрано мову програмування Python, тому що вона є простою у використанні, їй притаманна динамічна типізація і автоматичне управління пам’яттю і вона добре підходить для розробки систем машинного навчання. До того ж, Python підтримується на більшості платформ для роботи зі штучним інтелектом.

Проаналізуємо варіанти інструментів для практичної реалізації спроектованої системи. Фреймворк – програмна платформа, яка втілює рішення, що полегшують роботу над складними системами.

XGBoost – це фреймворк з відкритим вихідним кодом, який пропонує систему градієнтного покращення для C ++, Java, Python, R. Він розроблений для забезпечення високої ефективності, гнучкості та портативності. Після своєї презентації фреймворк і до сьогодні залишається фаворитом для вирішення більшості завдань. XGBoost фокусується на швидкості обчислень і продуктивності моделі і підходить для вирішення завдань регресії, класифікації та впорядкування. Коли є можливість представлення даних у вигляді таблиці, тоді і точність і продуктивність будуть помітно кращими, ніж у рішень з глибоким навчанням [30] У таблиці 3.2 наведено характеристику платформи XGBoost.

Таблиця 3.2 – характеристика платформи XGBoost

Переваги	Недоліки
Точність	Вузька спеціалізація
Швидкість і зручність, особливо для моделей типу «дерево рішень»	
Доцільно використовувати для перевірки гіпотез	

DistBelief – це платформа для навчання глибоких нейронних мереж, яка повністю уникає графічних процесорів і замість цього виконує паралельні обчислення з кластерами продуктових машин. DistBelief вперше був представлений у роботі «Широкомасштабні розподілені глибокі мережі» 2012 року. DistBelief в значній мірі спирається на асинхронне передавання повідомлень. Платформа дозволила працівникам Google будувати великі нейронні мережі та масштабувати навчання до тисяч ядер у своїх центрах обробки даних. Компанія Google використовувала його, щоб продемонструвати, що таким поняттям, як «кішка», можна навчитися з неіменованих зображень YouTube, покращити

розпізнавання людського мовлення в додатку Google на 25% та побудувати пошук зображень у Google Фото. DistBelief також підготував модель Inception, яка перемогла у масштабному виклику візуального розпізнавання Imagenet у 2014 році. Хоча DistBelief був дуже успішним, він мав деякі обмеження. Він був вузько орієнтований на нейронні мережі, його було важко налаштувати, і він був тісно пов'язаний із внутрішньою інфраструктурою Google – унеможливлюючи і без того складний зовнішній обмін кодами досліджень [31]. У таблиці 3.3 наведено характеристику платформи DistBelief.

Таблиця 3.3 – Характеристика платформи DistBelief

Переваги	Недоліки
Робота за принципом паралельних обчислень	Вузька спеціалізація
Можливість будувати масштабні нейронні мережі	Критична залежність від інфраструктури Google

TensorFlow – це потужний фреймворк, який функціонує шляхом реалізації ряду вузлів обробки, кожен з яких представляє математичну операцію, а весь ряд вузлів називається «графом». Спочатку задумувався розробниками як продовження закритої системи машинного навчання DistBelief, проте в процесі розробки компанія зробила вибір на користь користувачів і надала відкритий доступ до фреймворка. Завдяки цьому платформа постійно розвивається, адже багато ентузіастів працюють над покращенням існуючих рішень і створенням нових. TensorFlow складається з гнучкої екосистеми інструментів, бібліотек і ресурсів спільноти. Це дозволяє дослідникам створювати додатки з впровадженням технологій машинного навчання. Платформа надає інтуїтивно зрозумілі високорівневі API-інтерфейси зі швидким виконанням, що забезпечує майже негайну ітерацію моделі і простоту налагодження. Завдяки мультиплатформності дозволяє навчати і розгортати моделі в хмарному середовищі чи локально, незалежно від використовуваного користувачем мови.

Варто зазначити, що TensorFlow завжди забирає всю відеопам'ять. Для її обмеження необхідно створювати файл конфігурації і явно вказувати, що і скільки можна взяти. Через таку особливість можуть виникнути проблеми в роботі [32] У таблиці 3.4 наведено характеристику платформи Tensorflow.

Таблиця 3.4 – характеристика платформи Tensorflow

Переваги	Недоліки
Численна спільнота користувачів	Недружелюбність, система не вказує на помилки в коді
Оптимізація ресурсів для обчислень	Необхідність постійно контролювати використання пам’яті
Популярність і велика кількість існуючих рішень	Незадовільна документація

PyTorch – це середовище для машинного навчання на мові Python з відкритим вихідним кодом, що забезпечує тендерне обчислення з GPU-прискоренням. Було розроблене компанією Facebook. Фреймворк підходить для швидкого створення прототипів в дослідженнях, а також для любителів та невеликих проектів. Він надає динамічні графічні обчислення, які дозволяють опрацьовувати вводи та виводи перехідної тривалості, що корисно, наприклад, при роботі з рекурентними нейронними мережами. Навідміну від TensorFlow, PyTorch менш гнучкий в підтримці різних платформ. У ньому немає рідних інструментів для візуалізації даних, але є сторонній аналог з назвою tensorboardX [33]. У таблиці 3.5 наведено характеристику платформи PyTorch.

Таблиця 3.5 – Характеристика платформи PyTorch

Переваги	Недоліки
Нескладно створювати нові шари і працювати з GPU	Погана документація
Широкий вибір попередньо навчених моделей	Відсутність підтримки більшості платформ

На основі варіантного аналізу платформ машинного навчання, було обрано Tensorflow для створення програмного рішення з розпізнаванням жестів. Однією з головних ознак об’єктно-орієнтованого програмування є залежність кожного з об’єктів від конкретного класу. Клас виступає загальною характеристикою, чимось схожим на інженерне креслення, в якому описано основні ознаки. І уже на основі таких креслень створюються об’єкти класу. У роботі Tensorflow є як схожі, так і відмінні моменти з описаною вище залежністю. Фреймворк відділяє визначення обчислень від їх безпосереднього виконання: графи визначають операції, але операції виконуються тільки всередині сесій. Якщо розглядати граф як креслення, то сесія – середовище для інженерної діяльності, наприклад, верстат чи будівельний майданчик. При цьому графи і сесії створюються незалежно один від одного.

Для роботи із зображеннями було обрано бібліотеку OpenCV. OpenCV є найбільш відомою бібліотекою комп'ютерного зору, що має велику аудиторію користувачів, і по суті, задає стандарти в області машинного навчання, а саме в задачах розпізнавання образів. Вона має багато алгоритмів, що працюють «з коробки», має відкритий вихідний код. Бібліотека добре підтримується, оскільки має велику кількість користувачів. У інструментів є хороша і докладна документація, існує велика кількість матеріалів та описів тих чи інших функцій. Бібліотекою можна користуватися мовами C, C++, Python (а також Matlab, C#, Java) під різними операційними системами і це далеко не повний список переваг бібліотеки, що дозволяє зробити висновок, що OpenCV є найкращим вибором.

3.2 Розробка загального алгоритму розпізнавання жестів для керування комп’ютерними іграми

У процесі проектування розробимо алгоритм роботи системи розпізнавання жестів для керування у комп’ютерних іграх.

Крок 1: Запуск програми.

Крок 2: Режим очікування. Система прийматиме відеопотік, очікуючи появу рухів на фоні.

Крок 3: Після того, як користувач навів руку на робочу область і почав жестикулювати, алгоритм розпізнає і класифікує жести.

Крок 4: Якщо жест невідомий (його немає в базі даних або важко розпізнати), то алгоритм повертається до пункту 3.

Крок 5: Якщо жест відомий, система передає грі команду, що відповідає цьому жесту.

Крок 6: Розпізнавання відбувається до тих пір, поки користувач не вийде з програми.

Рисунок 3.2 – Схема алгоритму роботи програмного забезпечення розпізнавання жестів для керування у комп’ютерних іграх

3.3 Моделювання інтелектуальної системи розпізнавання жестів для керування комп'ютерними іграми із використанням мови UML

Програмний інтерфейс представляє собою серверну частину, де виконується обробка даних, зчитування відеопотоку, виведення відеозображення на екран користувача, передача результатів роботи нейронної мережі в інтерпретатор команд.

У блоці розпізнавання працює згорткова нейромережа, створена за принципами каскадів Хаара. Тут відстежуються координати положення руки та розпізнаються жести, отримані дані передаються програмному інтерфейсу. На невідомі жести система ніяк не реагує.

Інтерпретатор команд отримує інформацію про класифіковані жести та співставляє їх з командами керування. Залежно від отриманих координат положення руки жести можуть інтерпретуватися по-різному. Наприклад, на рисунку 3.3 зображено взаємодію блока розпізнавання жестів з грою «Dinosaur Game». Для керування у грі потрібно дві команди – «вверх і вниз».

Отримана інформація про жести надходить до інтерпретатора команд. Залежно від позиції руки в межах видимої для веб-камери області, а також від того, закрита кисть руки чи відкрита, виконуватимуться відповідні команди.

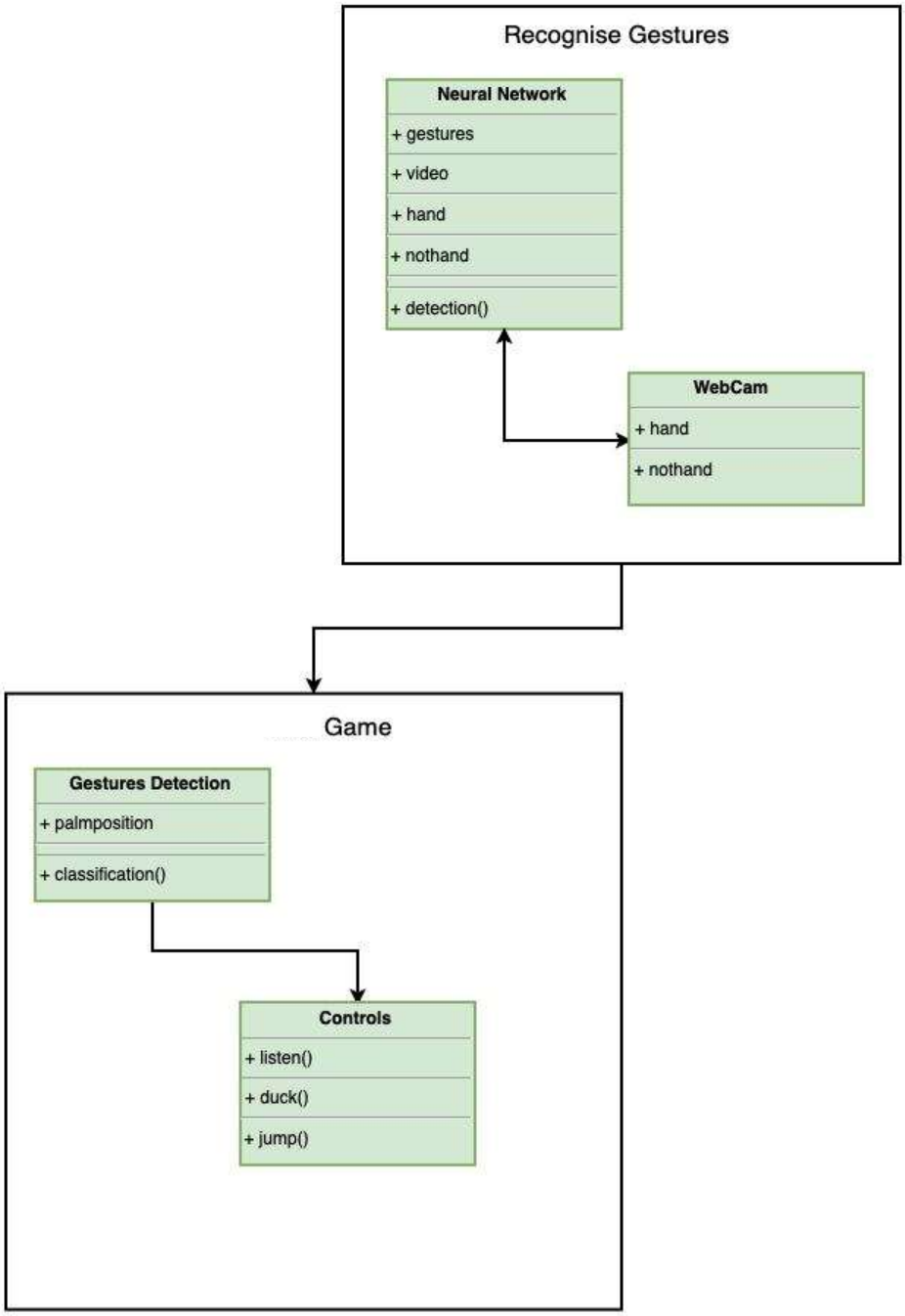


Рисунок 3.3 – UML-діаграма компонентів системи розпізнавання жестів для керування у комп’ютерних іграх

3.4 Розробка програмного забезпечення для розпізнавання жестів

Для початку роботи створено віртуальне середовище за допомогою інструменту Anaconda – дистрибутиву для машинного навчання на Python.

Для виконання ефективних обчислень в Python використовується бібліотека NumPy, яка здійснює складні операції, наприклад перемноження матриць, за межами цієї мови, використовуючи найбільш ефективний код, реалізований іншою мовою [34]. Однак, виникає додаткове навантаження при перемиканні назад до Python після кожної операції. Тим не менш, це всеодно набагато швидше ніж обраховувати кожен піксель за допомогою базових алгоритмів.

Для процесу попередньої обробки зображень та подальшої класифікації жестів було розроблено нейромережеву версію методу ознак Хаара. Це дозволило збільшити швидкодію розпізнавання на 12% у порівнянні зі звичайним алгоритмом. Фрагмент коду, що реалізує роботу каскаду класифікаторів:

```
inline __m128 ValidSqrt(__m128 value)
{
  return _mm_blendv_ps(_mm_set1_ps(1.0f), _mm_sqrt_ps(value),
_mm_cmpgt_ps(value, _mm_set1_ps(0.0f)));
}

inline __m128i Sum32ip(uint32_t * const ptr[4], size_t offset)
{
  __m128i s0 = _mm_loadu_si128((__m128i*)(ptr[0] + offset));
  __m128i s1 = _mm_loadu_si128((__m128i*)(ptr[1] + offset));
  __m128i s2 = _mm_loadu_si128((__m128i*)(ptr[2] + offset));
  __m128i s3 = _mm_loadu_si128((__m128i*)(ptr[3] + offset));
  return _mm_sub_epi32(_mm_sub_epi32(s0, s1), _mm_sub_epi32(s2,
s3));
}

inline __m128 Norm32fp(const HidHaarCascade & hid, size_t
offset)
{
  __m128 area = _mm_set1_ps(hid.windowArea);
  __m128 sum = _mm_cvtepi32_ps(Sum32ip(hid.p, offset));
```

```
  __m128 sqsum = _mm_cvtepi32_ps(Sum32ip(hid.pq, offset));
  return ValidSqrt(_mm_sub_ps(_mm_mul_ps(sqsum, area),
_mm_mul_ps(sum, sum)));
}
```

За завданням технологія повинна коректно працювати з будь-якою сучасною веб-камерою. Більшість нових веб-камер відображають відеопоток в HD-якості, що еквівалентно розмірності 1280x720 пікселів.

Частина коду програмного забезпечування, що реалізує розпізнавання жестів:

```
    def is_in_triangle(point, triangle):
        # barycentric coordinate system
        x, y = point
        (xa, ya), (xb, yb), (xc, yc) = triangle
        a = ((yb - yc)*(x - xc) + (xc - xb)*(y - yc)) / ((yb -
yc)*(xa - xc) + (xc - xb)*(ya - yc))
        b = ((yc - ya) * (x - xc) + (xa - xc) * (y - yc)) /
((yb - yc) * (xa - xc) + (xc - xb) * (ya - yc))
        c = 1 - a - b
        if 0 <= a <= 1 and 0 <= b <= 1 and 0 <= c <= 1:
            return True
        else:
            return False
    def load_graph(path):
        detection_graph = tf.Graph()
        with detection_graph.as_default():
            graph_def = tf.GraphDef()
            with tf.gfile.GFile(path, 'rb') as fid:
                graph_def.ParseFromString(fid.read())
                tf.import_graph_def(graph_def, name='')
            sess = tf.Session(graph=detection_graph)
        return detection_graph, sess
    def detect_hands(image, graph, sess):
        input_image =
graph.get_tensor_by_name('image_tensor:0')
        detection_boxes =
graph.get_tensor_by_name('detection_boxes:0')
        detection_scores =
graph.get_tensor_by_name('detection_scores:0')
        detection_classes =
graph.get_tensor_by_name('detection_classes:0')
```

```
        image = image[None, :, :, :]
        boxes, scores, classes = sess.run([detection_boxes,
detection_scores, detection_classes],
        feed_dict={input_image: image})
        return np.squeeze(boxes), np.squeeze(scores),
np.squeeze(classes)
```

Для початку, було розроблено програмний код для переведення жестового керування у «Dinosaur Game» – грі від Google, в яку браузер пропонує зіграти, коли втрачається зв’язок з інтернетом. Для керування у цій грі потрібно всього дві команди – «стрибок» і «присідання». Щоб реалізувати процес керування грою за допомогою жестів, вікно з зображенням камери буде поділено на дві частини – верхня для стрибків, нижня – для присідань.

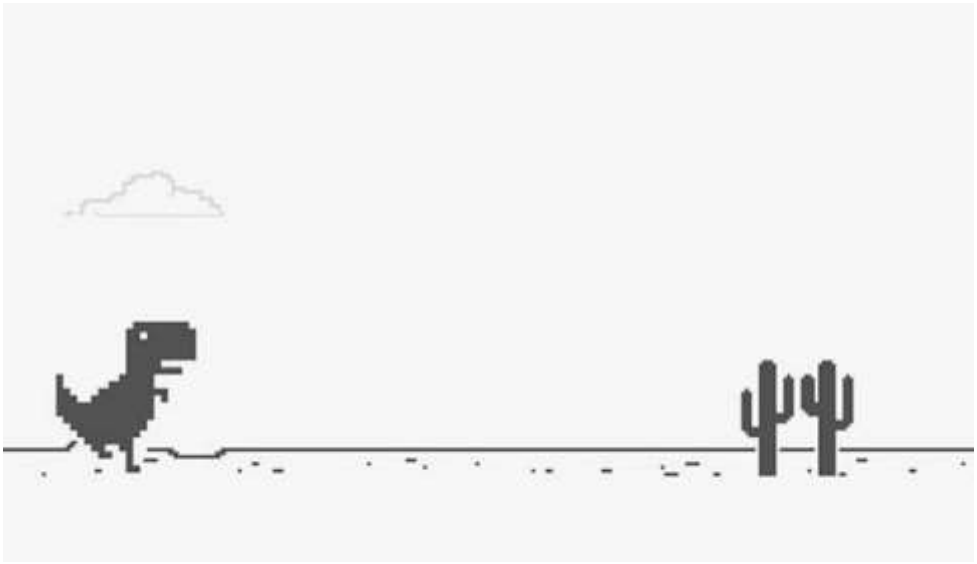


Рисунок 3.4 – Загальний вигляд інтерфейсного вікна гри «Dinosaur Game»

Фрагмент коду, що реалізує керування у грі «Dinosaur Game»:

```
def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
```

```
        mp = _mp.get_context("spawn")
        v = mp.Value('i', 0)
        lock = mp.Lock()
        process = mp.Process(target=dinosaur, args=(v, lock))
        process.start()
        while True:
            key = cv2.waitKey(10)
            if key == ord("q"):
            break _, frame = cap.read()
            frame = cv2.flip(frame, 1)
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            boxes, scores, classes = detect_hands(frame, graph,
sess)
            frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
            results = predict(boxes, scores, classes,
FLAGS.threshold, FLAGS.width, FLAGS.height)
            if len(results) == 1:
                x_min, x_max, y_min, y_max, category =
results[0]
                x = int((x_min + x_max) / 2)
                y = int((y_min + y_max) / 2)
                cv2.circle(frame, (x, y), 5, RED, -1)
                if category == "Closed":
                    action = 0  # Do nothing
                    text = "Run"
                elif category == "Open" and y < FLAGS.height/2:
                    action = 1 # Jump
                    text = "Jump"
                elif category == "Open" and y > FLAGS.height/2:
                    action = 2
                    text = "Duck"
                else: action = 0
                    text = "Run"
                with lock:
                    v.value = action
                cv2.putText(frame, "{}".format(text), (x_min,
y_min - 5),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)
            overlay = frame.copy()
            cv2.rectangle(overlay, (0, 0), (FLAGS.width,
int(FLAGS.height / 2)), YELLOW, -1)
               cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 -
FLAGS.alpha, 0, frame)
```

```
            cv2.imshow('Detection', frame)
        cap.release() cv2.destroyAllWindows()
```

Для реалізації можливостей системи жестові команди адаптовано для керування у класичних та сучасних двовимірних платформерах. Гра-платформер – жанр комп'ютерних ігор, ігровий процес в яких складається зі стрибків персонажа по різноманітних платформах (звідси і назва) та через перешкоди, збирання предметів, звичайно необхідних для завершення рівня. У традиційних двовимірних платформерах персонаж рухається зліва направо, деякі ігри дозволяють рухатися у будь-якому напрямку. У більшості платформерів для боротьби з ворогами використовується стрибок, тобто для перемоги потрібно впасти на противника зверху. Однак є представники жанру, у яких реалізовано бойову систему та стрільбу.

Отже, для комфортного керування у будь-якому традиційному платформері потрібні 6 команд – «рух вліво», «рух вправо», «стрибок», «присідання», «постріл» та «меню паузи».

Для реалізації ігрового процесу з використанням жестів зображення з веб-камери буде уявно поділено на три вертикальні частини при розпізнаванні: ліва частина для руху вліво, права – для руху вправо, середня частина ж відповідатиме за стояння на місці. Стрибок, удар, постріл і вихід до меню паузи можна буде виконати у будь-якій області екрану.

Фрагмент коду, що реалізує керування у грі-платформері:

```
def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
    mp = _mp.get_context("spawn")
    v = mp.Value('i', 0)
    lock = mp.Lock()
    process = mp.Process(target=sprite, args=(v, lock))
```

```
    process.start()
    while True:
        key = cv2.waitKey(10)
        if key == ord("q"):
            break _, frame = cap.read()
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        boxes, scores, classes = detect_hands(frame, graph,
sess)
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        results = predict(boxes, scores, classes,
FLAGS.threshold, FLAGS.width, FLAGS.height)
        if len(results) == 1:
            x_min, x_max, y_min, y_max, category = results[0]
            x = int((x_min + x_max) / 2)
            y = int((y_min + y_max) / 2)
            cv2.circle(frame, (x, y), 5, RED, -1)
            if category == "Open" and x <= FLAGS.width / 3:
                action = 7  # Left jump
                text = "Jump left"
            elif category == "Closed" and x <= FLAGS.width / 3:
                action = 6  # Left
                text = "Run left"
            elif category == "Open" and FLAGS.width / 3 < x <= 2
* FLAGS.width / 3: action = 5  # Jump
                text = "Jump"
            elif category == "Closed" and FLAGS.width / 3 < x <=
2 * FLAGS.width / 3: action = 0  # Do nothing
                text = "Stay"
          elif category == "Open" and x > 2 * FLAGS.width / 3:
                action = 2  # Right jump
                text = "Jump right"
          elif category == "Closed" and x > 2 * FLAGS.width / 3:
                action = 1  # Right
                text = "Run right"
          else: action = 0
                text = "Stay"
          with lock: v.value = action
 cv2.putText(frame, "{}".format(text), (x_min, y_min – 5),
 cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)
 cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 – FLAGS.alpha,
0, frame)
    cv2.imshow('Detection', frame) cap.release()
```

3.5 Тестування розробленого програмного забезпечення та аналіз результатів його роботи

У процесі запуску програмного забезпечення, на екрані відображається вікно передачі зображення з камери, після чого гравець отримує доступ до жестового керування грою. Тестування проведено з використанням зображення звичайної HD-камери з відеорядом розмірністю 1280×720 пікселів. Першим буде протестовано керування у грі «Dinosaur Game». Для старту потрібно віддати команду «стрибок», піднявши кулак у верхню зону камери та розтиснувши долоню.

Головне завдання в процесі гри – ухилятись від перешкод та набрати якомога більше очок. Для успішного керування динозавром, потрібно користуватись двома командами – «стрибок» і «присідання». Вікно, у якому відображається камера, поділене на дві горизонтальні частини. Верхня зона вікна відповідає за стрибки, а нижня, відповідно, за присідання. Керування здійснюється одною рукою. У стані спокою і при переміщенні між зонами пальці руки повинні бути зжаті в кулак. Щоб здійснити стрибок, потрібно перевести руку у верхню зону та розкрити кисть, відпустити пальці та направити їх догори [Рисунок 3.5].

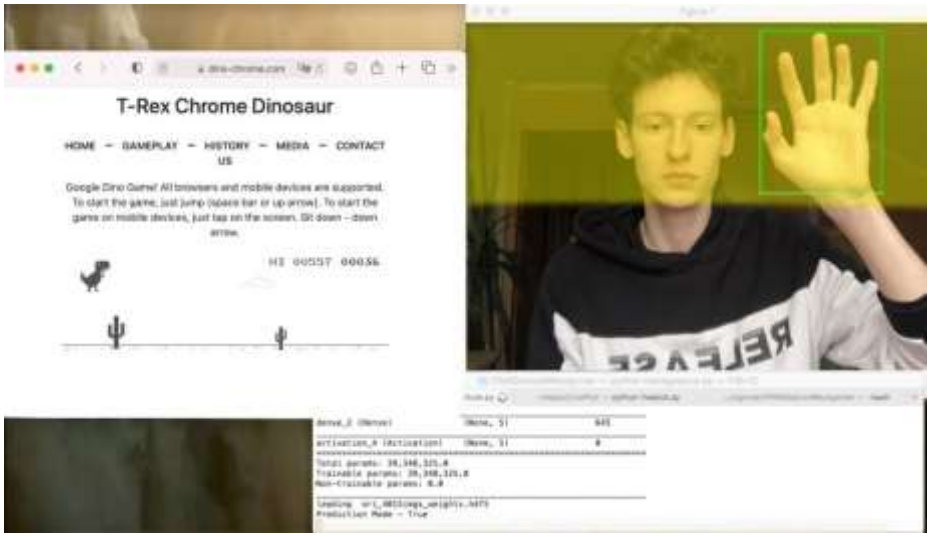


Рисунок 3.5 – Загальний вигляд інтерфейсного вікна під час виконання команди «стрибок» у грі «Dinosaur Game»

На рисунку 3.6 зображено виконання команди «Присідання».

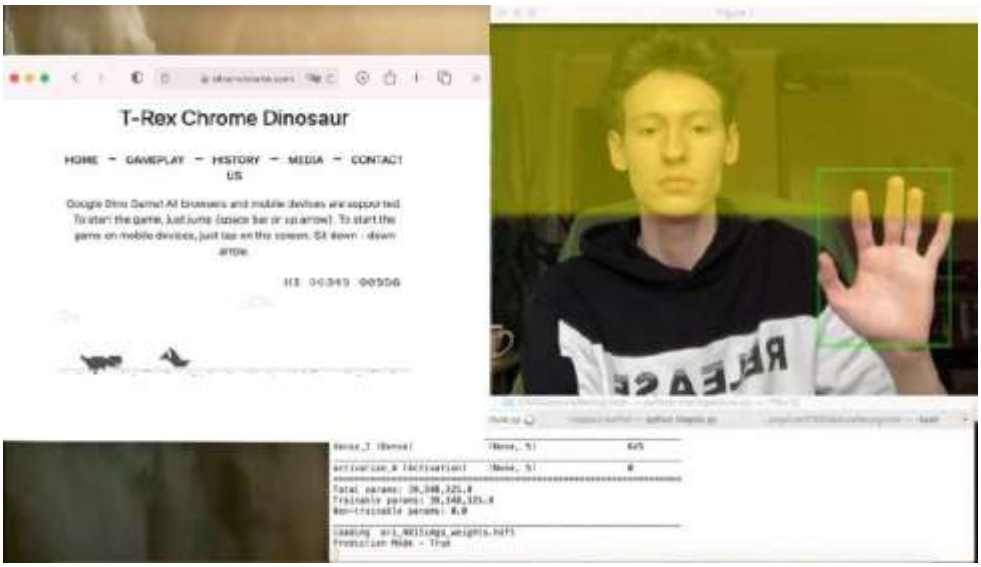


Рисунок 3.6 – Загальний вигляд інтерфейсного вікна під час виконання команди «присідання» у грі «Dinosaur Game»

Як і задумано правилами, коли гравець не реагує або не встигає зреагувати на перешкоду, гра закінчується [Рисунок 3.7].

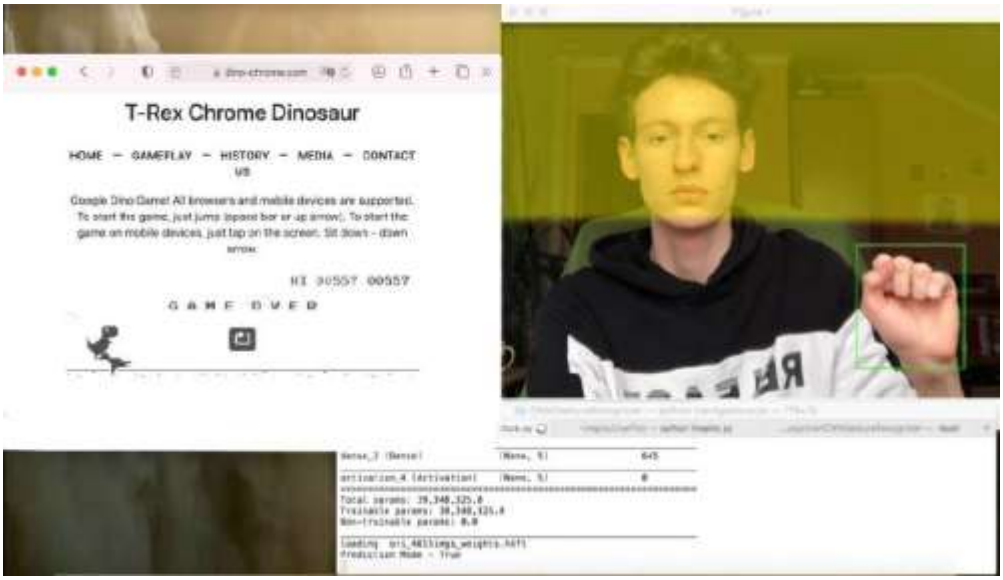


Рисунок 3.7 – Загальний вигляд інтерфейсного вікна програшу у грі «Dinosaur Game»

Розроблено також інтерпретацію жестів у додаткові команди для того, аби зробити керування у іграх-платформерах універсальним. У платформерах зазвичай головне завдання – пройти рівень від початку до кінця. На шляху до цілі гравцеві трапляються перешкоди, корисні предмети та штучний інтелект у вигляді ворогів. Серія ігор «Sonic» про пригоди їжака Соніка та його друзів – типовий представник жанру. Для керування використовуються команди «вліво», «вправо», «стрибок» та «присідання».

Тестування керування у описаних нижче іграх проводилось на сайті playclassic.games – онлайн-порталі, на якому розміщена велика кількість класичних ігор восьмидесятих, дев’яностих та двохтисячних років для ігрових приставок та персональних комп’ютерів [35]. У всі ці ігри можна пограти прямо в браузері, нічого завантажувати не потрібно.

На рисунку 3.8 представлено приклад роботи технології керування жестами у грі «Sonic & Knuckles».

Рисунок 3.8 – Загальний вигляд вікна завантаження гри «Sonic & Knuckles»

Головний герой може рухатись вліво і вправо, стрибати і боротись з ворогами, падаючи на них зверху. Для жестового керування грою вікно з виведенням зображення камери поділено на три вертикальні зони. Ліва зона – для руху вліво, права – для руху в право, а між ними зона для стану спокою. Щоб почати рух вправо, потрібно перевести закриту руку у праву зону. На рисунку 3.9 зображено приклад виконання руху вправо.

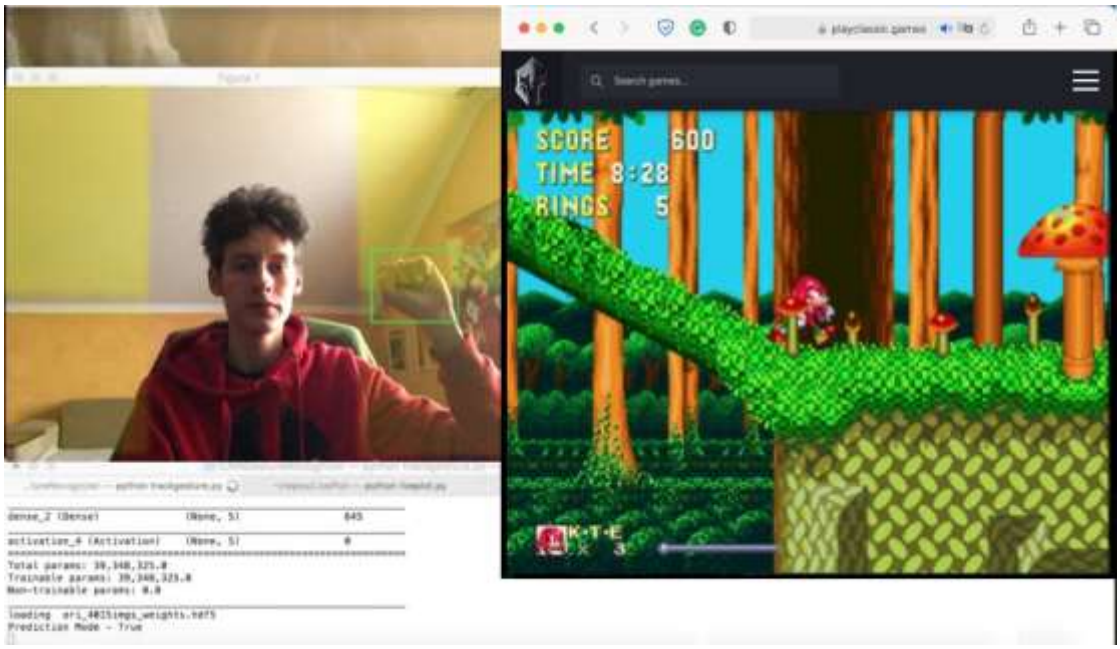


Рисунок 3.9 – Загальний вигляд інтерфейсного вікна при виконанні руху вправо у грі «Sonic & Knuckles»

Щоб рухатись вліво, потрібно перевести закриту руку в ліву зону. На рисунку 3.10 зображено приклад виконання руху вправо.

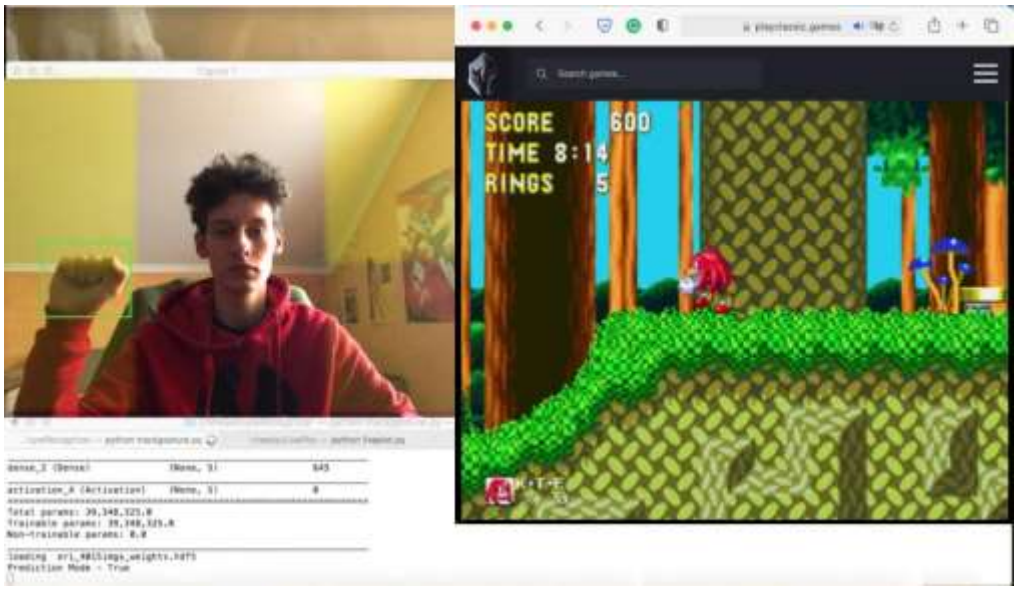


Рисунок 3.10 – Загальний вигляд інтерфейсного вікна при виконанні руху вліво у грі «Sonic & Knuckles»

Для стрибка у грі потрібно розкрити руку у будь-якому місці. На рисунку 3.11 зображено приклад виконання стрибка, у цьому випадку – стрибка вліво.

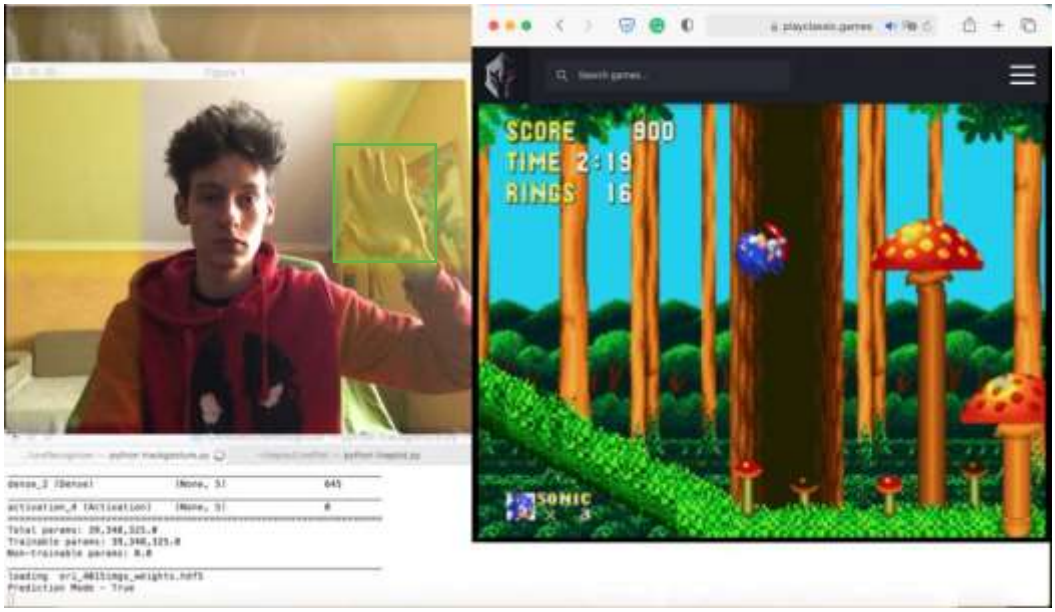


Рисунок 3.11 – Загальний вигляд інтерфейсного вікна при виконанні команди «стрибок» у грі «Sonic & Knuckles»

Для перевірки роботи команд «удар» та «пауза» проведено тестування системи у грі «Черепашки-Ніндзя: Падіння Клану Фут». У грі потрібно керувати однією з черепашок на вибір, перемагати ворогів та отримувати за це очки. Рухи вправо, вліво та «стрибок» працюють аналогічно до попередньої гри. На рисунку 3.12 наведено приклад виконання руху вправо та вліво відповідно у грі «Черепашки-Ніндзя: Падіння Клану Фут».

Рисунок 3.12 – Загальний вигляд інтерфейсного вікна при виконанні рухів вправо та вліво у грі «Черепашки-Ніндзя: Падіння Клану Фут»

Щоб поставити гру на паузу, потрібно вказати жест «два», демонструючи відкриті вказівний та середній пальці, роздільно. На рисунку 3.13 наведено приклад виконання команди «пауза».

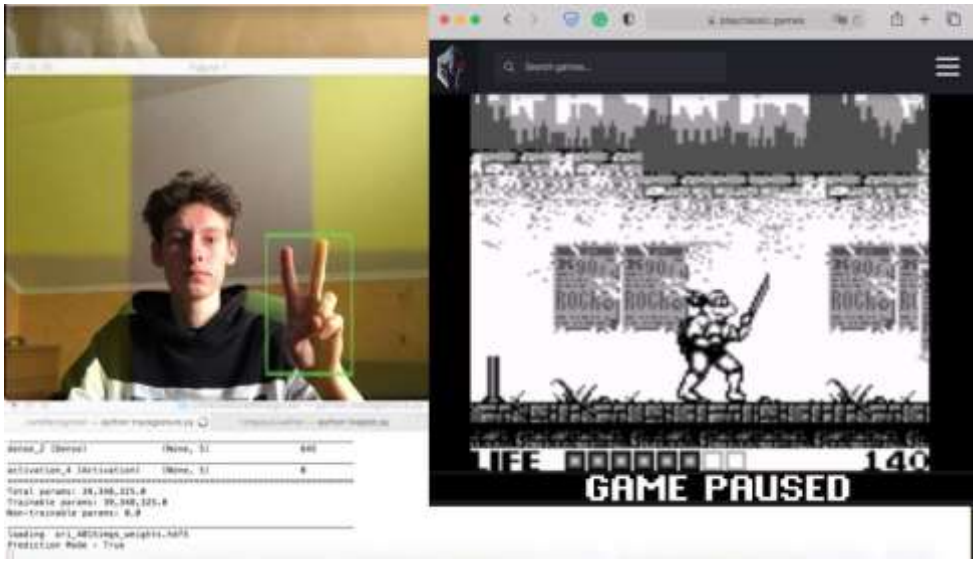


Рисунок 3.13 – Загальний вигляд інтерфейсного вікна при виконанні руху вправо у грі «Черепашки-Ніндзя: Падіння Клану Фут»

Для нанесення удару необхідно вказати жест «один», розкривши вказівний палець. На рисунку 3.14 наведено приклад виконання команди «удар».

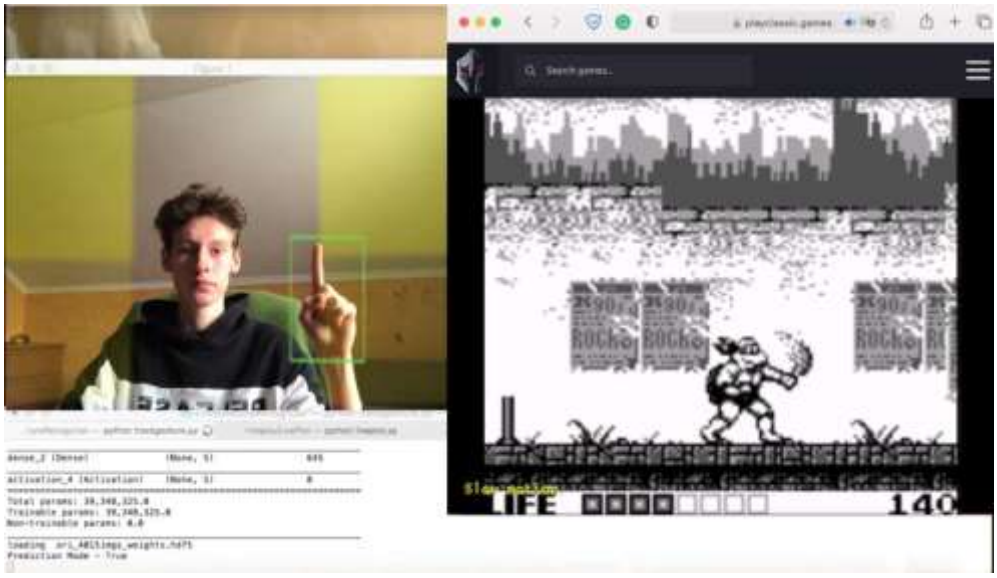


Рисунок 3.14 – Загальний вигляд інтерфейсного вікна при виконанні руху вліво та стрибка у грі «Черепашки-Ніндзя: Падіння Клану Фут»

Система повинна працювати з камерою, що видає зображення у якості HD, що рівне розмірності 1280×720 пікселів, а точність розпізнавання має бути не нижче 95%. Спершу варто перевірити, як розпізнавання жестів відбуватиметься при гіршій якості зображення. Для наступного тесту відеопотік з камери було знижено до якості 240p, що еквівалентно розширенню 320×240 пікселів. У сфері потокового передавання 240p класифікується як «відеопотік низької чіткості». Тестування показало, що зниження рекомендованої якості вхідного потоку не перешкоджає процесу розпізнавання. Завдяки влучній комбінації методів обробки зображень та методу Хаар-подібних характеристик жестове керування можливе навіть при низькій чіткості відеоряду з камери. На рисунку 3.15 зображено приклад виконання команди «присідання» при якості відео рівній 240p.

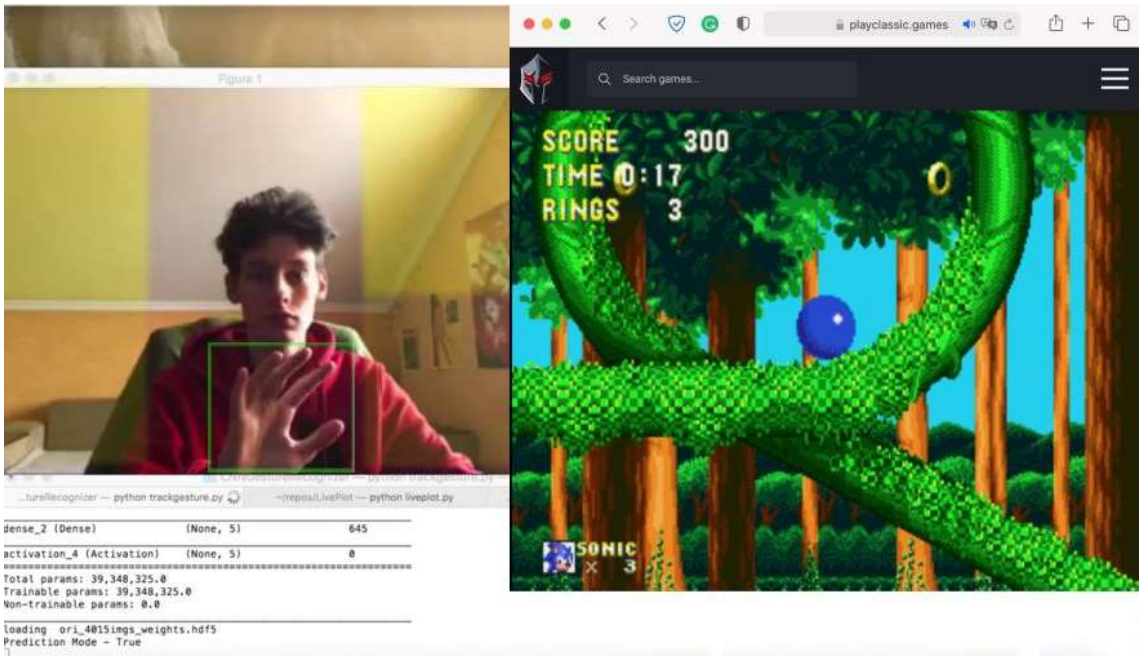


Рисунок 3.15 – Загальний вигляд інтерфейсного вікна в умовах низької чіткості зображення

Здійснено перевірку роботи системи при слабкому освітленні. Для цього було уникнено всіх джерел освітлення, а єдиним джерелом, що залишилось, став екран комп’ютера (рис. 3.16):

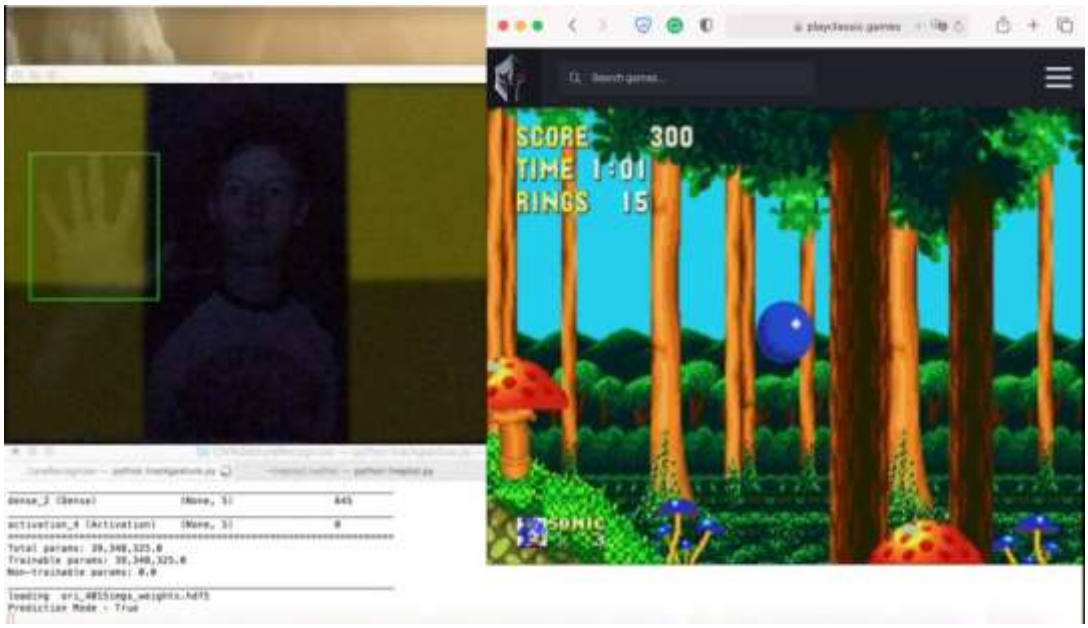


Рисунок 3.16 – Загальний вигляд інтерфейсного вікна жестового керування в умовах поганого освітлення

Експеримент показав, що нейромережа все ще розпізнає деякі жести, однак команди керування часто спрацьовують некоректно, тому очевидно, що система потребує якісного освітлення і не працює при низькому рівні освітлення.

Здійснено перевірку точності розпізнавання жестів. Для цього в процесі гри при нормальних умовах виконано по 50 повторень таких жестових команд, як: «рух вліво», «рух вправо», «пауза», «постріл». Виконано по 100 повторень команд «стрибок» і «присідання». Результати тестування представлено в таблиці 3.6.

Таблиця 3.6 – Результати тестування точності розпізнавання жестів за допомогою розробленого програмного забезпечення

Виконання команди	Кількість спроб	Правильно розпізнано	Точність розпізнавання, %
Рух вліво	50	48	96
Рух вправо	50	49	98
Стрибок	100	95	95
Присідання	100	94	94
Пауза	50	50	100
Удар	50	45	90

За результатами тестування у нормальних умовах визначено середню точність розпізнавання. Для цього суму точності у відсотках розділено на кількість тестів. Середня точність розпізнавання жестів – 95,5%, що більше ніж 95%, а отже мета роботи досягнута.

Варто звернути увагу на аналогічну реалізацію – Hand Gesture Recognition (HGR) using Deep Learning in Matlab. Це програмне забезпечення дозволяє застосовувати технологію розпізнавання жестів у різних задачах, в тому числі і для керування у комп’ютерних іграх. Детекцію та класифікацію жестів у системі виконує ConvNet – згорткова нейромережа, яка не використовує ніякі додаткові ознаки в процесі розпізнавання. У зв’язку з цим, система є чутливою до повертання руки, а кутові відхилення можуть нашкодити процесу розпізнавання. У таблиці 3.7 наведено результати тестування HGR using Deep Learning in Matlab.

Таблиця 3.7 – Результати тестування точності розпізнавання жестів

Виконання команди	Кількість спроб	Правильно розпізнано	Точність розпізнавання, %
Стрибок	100	92	92
Постріл	100	94	94
Пауза	100	92	92

За результатами точність розпізнавання склала 92,6%.

Комп’ютерні ігри досить динамічні, тому користувачеві може бути зручно незначно нахиляти руку у процесі керування, частіше за все це стається непомітно для самого користувача. Розроблене програмне забезпечення дозволяє компенсувати ці незначні нахили і викривлення за рахунок застосування каскадів Хаар-подібних ознак, тому ігровий процес не порушується і розпізнавання працює безперебійно із кутом нахилу руки до 15 градусів.

3.6 Висновок до розділу 3

Розроблено загальний алгоритм функціонування інтелектуальної системи розпізнавання жестів для керування комп’ютерними іграми. Вона складається з чотирьох основних компонентів: програмного інтерфейсу, блоку розпізнавання, інтерпретатора рухів у команди керування грою та сховища у вигляді масива даних.

Розроблено структуру інформаційної технології, алгоритм роботи та UML-модель системи розпізнавання жестів для керування у комп’ютерних іграх. На основі результатів проектування здійснено програмну реалізацію компонентів системи.

Виконано тестування роботи розробленого програмного забезпечення. У процесі тестування визначено, що система коректно працює в умовах чіткості зображення нижчій за рекомендовану. Досягнуто точності розпізнавання жестів у 95,5%, що задовільняє поставлені вимоги.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту інформаційна технологія розпізнавання жестів для керування комп'ютерними іграми. Особливістю програми є те, що дана технологія надає гравцеві унікального ігрового досвіду, так як користувач грає у гру без клавіатури, за допомогою жестів руки, які показує в камеру.

Аналогом розробки є Microsoft Kinnect за ціною 2000 грн, HGR using Deep Learning in Matlab за ціною 2700 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 4.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	3	3	4
Цінова політика	3	4	3
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	3	3
Супровідна документація	3	3	4
Сума	41	40	42
Середньоарифметична сума балів	(41+40+42) / 3 = 41		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, так як чим більше участі в процесі бере гравець – тим цікавіше йому грати. Із технічного забезпечення потрібна лише веб-камера, що дасть змогу застосовувати продукт без придбання додаткового обладнання високої вартості, а тому зробить отримання нового ігрового досвіду доступнішим (кожен сучасний ноутбук має веб-камеру). Розроблений програмний продукт відрізняється від існуючих тим, що дана технологія надає гравцеві унікальний ігровий досвід, так як користувач грає у гру без клавіатури, за допомогою жестів руки, які показує в камеру.

4.2 Прогнозування витрат на виконання науково-дослідної та дослідно-конструкторської роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \,, \tag{4.1}$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$Т_р$ – число робочих днів в місяці, 23 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.4.

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	27000	1173,91	46	54000,000
Інженер	22000	956,52	46	44000,000
Всього				98000,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 10,5 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 10,5\ \% / 100\ \% \qquad (4.2)$$

$З_д = (98000,00 \cdot 10,5\ \% / 100\ \%) = 10290,00$ (грн.)

Нарахування на заробітну плату розробників.

Згідно з чинним законодавством нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$Н_з$ = (98000,00 + 10290,00) · 22 % / 100 % = 23 823,80 (грн.)

Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючі, то витрати на матеріали і комплектуючі дорівнюють нулю.

У спрощеному вигляді амортизація обладнання, що використовувалась для розробки, розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 2,00 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{2,0}{12} = 1666,67 \ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.2. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$A_{н.р.} = Ц_{н.р.} * Н_а * \frac{t_{вик.}}{12} \quad (4.5)$$

Але, так як вартість ліцензійної ОС (Windows 11 Professional Retail) та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 530 грн.

Таблиця 4.5 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна переферія	20000	2	2,00	1666,667
Офісне обладнення	20000	4	2,00	833,333
Приміщення	830000	20	2,00	6916,667
Всього				9416,67

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,6 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,6 \cdot 8 \cdot 46 \cdot 6,2 = 1232,064$ (грн.)

Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в = 98000,00 * 125\% / 100\% = 122500$ (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$H_{нзв}$ = 98000,00 * 125 % / 100 % = 122500 (грн.)

Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ = 98000,00+10290,00+23823,80+9416,67+530+1232,06+122500+
+122500 = 388292,53 грн.

Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \quad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 388292,53 / 0,5 = 776585 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);

– терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.10)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

ρ – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 1200 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її

вдосконалення, можна буде підняти її ціну на 150 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 20000 шт., протягом другого року – на 30000 шт., протягом третього року на 35000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (0*150 + (1200 + 150)*20000)* 0,8333* 0,15) * (1 - 0,18) = 2459999,902 грн.

$\Delta П_2$ = (0*150 + (1200 + 150)*(20000+30000)* 0,8333* 0,15) * (1 - 0,18) = 6918749,723 грн.

$\Delta П_3$ = (0*150 + (1200 + 150)*(20000+30000+35000)* 0,8333* 0,15) * (1 - 0,18) = 11761874,530 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 21140624,15 грн.

Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= $(2459999,902/(1+0,1)^1)+(6918749,723/(1+0,1)^2)+(11761874,530/(1+0,1)^3)$ = 2236363,55 + 5717974,978 + 8836870,42 = 16791208,95 грн.

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 776585 = 1553170,12 \text{ грн.}$$

Тоді абсолютний економічний ефект Еабс або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП – PV, \quad (4.13)$$

$Е_{абс}$ = 16791208,95 – 1553170,12 = 15238038,82 грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра-хувати внутрішню економічну дохідність або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.14}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 15238038{,}82/1553170{,}12} - 1 = 1{,}211$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (4.16)$$

$$T_{ок} = 1 / 1{,}211 = 0{,}83 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,83 роки, то фінансування даної наукової розробки є доцільним.

4.4 Висновок до розділу 4

Економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 776585 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,83 роки.

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи розв’язано задачу розробки технології розпізнавання жестів для керування у комп’ютерних іграх.

У першому розділі магістерської роботи було виконано аналіз предметної області розпізнавання жестів для керування у комп’ютерних іграх, а саме – сформульовано постановку задачі, проведено огляд відомих методів вирішення задачі розпізнавання жестів для керування у комп’ютерних іграх. Також було здійснено аналіз програмних засобів розпізнавання жестів та визначено їх основні недоліки.

У другому розділі проаналізовано шляхи вирішення поставленої задачі. Обгрунтовано вибір методів реалізації інформаційної системи розпізнавання жестів для керування у комп’ютерних іграх. Розроблено загальний алгоритм, а також структуру інформаційної технології розпізнавання жестів для керування у комп’ютерних іграх. Інформаційна технологія передбачає виконання п’яти етапів, а саме: зчитування потоку з веб-камери, бінаризацію зображення за методом інтегрального представлення, виділення контурів руки за допомогою Хаар-подібних ознак, класифікацію жеста та виконання відповідної команди керування у комп’ютерній грі. Наведено UML-діаграму взаємодії компонентів системи. Для підвищення точності розпізнавання застосовано інтегральне представлення зображення та розроблено подання методу Хаар-подібних характеристик у вигляді нейромережі.

У третьому розділі розроблено програмне забезпечення, що виконує розпізнавання жестів для керування у комп’ютерних іграх, засобами мови програмування Python із застосуванням бібліотек OpenCV, Tensorflow, середовища Anaconda. Було здійснено тестування розробленої програми, при рекомендованій розмірності відеоряду у 1280×720 досягнуто точності розпізнавання у діапазоні 95-97%. Аналіз результатів тестування показав, що програма повністю виконує усі

поставлені задачі, що забезпечують підвищення точності розпізнавання жестів у неперервному відеопотоці.

У четвертому розділі, що є економічною частиною, виконано розрахунок витрат на розробку нового програмного продукту, сума яких складає 776585 гривень. Спрогнозовано орієнтовану величину витрат по кожній зі статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення. Розраховано період окупності витрат для інвестора та економічний ефект під час використання даної розробки. У результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,83 роки.

Таким чином, усі задачі магістерської кваліфікаційної роботи розв’язано, а отже мету досягнено.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шевчук А. В., Арсенюк І. Р. Застосування інформаційної технології розпізнавання жестів для керування у комп'ютерних іграх. // Scientific progress: innovations, achievements and prospects. Proceedings of the 3rd International scientific and practical conference. MDPC Publishing. Munich, Germany. 2022. Pp. 164 – 169. [Електронний ресурс] – Режим доступу: https://sci-conf.com.ua/iii-mizhnarodna-naukovo-praktichna-konferentsiya-scientific-progress-innovations-achievements-and-prospects-4-6-12-2022-myunhen-nimechchina-arhiv/
2. Дослідження аналітиків DFC Intelligence [Електронний ресурс] – Режим доступу: https://itc.ua/news/dfc-intelligence-chislo-igrayushhih-vvideoigry-lyudej-v-mire-prevysilo-3-mlrd-polovina-iz-nih-igraet-
3. Analysis of movement and gesture recognition using Leap Motion Controller [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/pii/S1877050918307403
4. Myo Gestures Control [Електронний ресурс] – Режим доступу: https://uncrate.com/myo-gesture-control-armband/
5. Nintendo Wii [Електронний ресурс] – Режим доступу: https://www.britannica.com/topic/Nintendo-Wii
6. Introducing PlayStation Move [Електронний ресурс] – Режим доступу: https://blog.playstation.com/archive/2010/03/10/introducing-playstationmove/
7. Tales In Tech History: Microsoft Kinect [Електронний ресурс] – Режим доступу: https://www.silicon.co.uk/e-innovation/microsoft-kinect-history-226781
8. Kinect Sign Language Translator Expands Communication Possibilities for the Deaf [Електронний ресурс] – Режим доступу: https://www.audiologyonline.com/releases/kinect-sign-language-translatorexpands-12288

9. Kai: Gestures Controller [Електронний ресурс] - Режим доступу: https://www.indiegogo.com/projects/kai-gesture-controller-buy-now-at-vicara-co#/
10. Oculus Touch [Електронний ресурс] - Режим доступу: https://developer.oculus.com/documentation/native/pc/dg-input-touch-overview/
11. Tap Strap2 [Електронний ресурс] - Режим доступу: https://www.tapwithus.com/product/tap-strap-2/
12. Motion Sensor Kit [Електронний ресурс] - Режим доступу: https://help.kano.me/hc/en-us/articles/360000785639-General-FAQs-Motion-Sensor-Kit
13. HGR using Deep Learning in MatLab [Електронний ресурс] - Режим доступу: https://www.pantechsolutions.net/hand-gesture-recognition-using-deep-learning-in-matlab
14. Навчання нейромережі з учителем, без вчителя, з підкріпленням — у чому відмінність? [Електронний ресурс] – Режим доступу: https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-spodkrepleniem/
15. Hand Gesture Recognition using Deep Learning in Matlab [Електронний ресурс] – Режим доступу: https://www.pantechsolutions.net/hand-gesture-recognition-using-deep-learning-in-matlab
16. Neural Network Models for Combined Classification and Regression [Електронний ресурс] – Режим доступу: https://machinelearningmastery.com/neural-network-models-for-combinedclassification-and-regression/
17. Модель персептрона [Електронний ресурс]- Режим доступу: http://dspace.nbuv.gov.ua/handle/123456789/131626?show=full
18. Recurrent Neural Networks [Електронний ресурс] – Режим доступу: https://www.ibm.com/cloud/learn/recurrent-neural-networks
19. Identify the cells' nuclei based on the deep learning neural network [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1911.09830

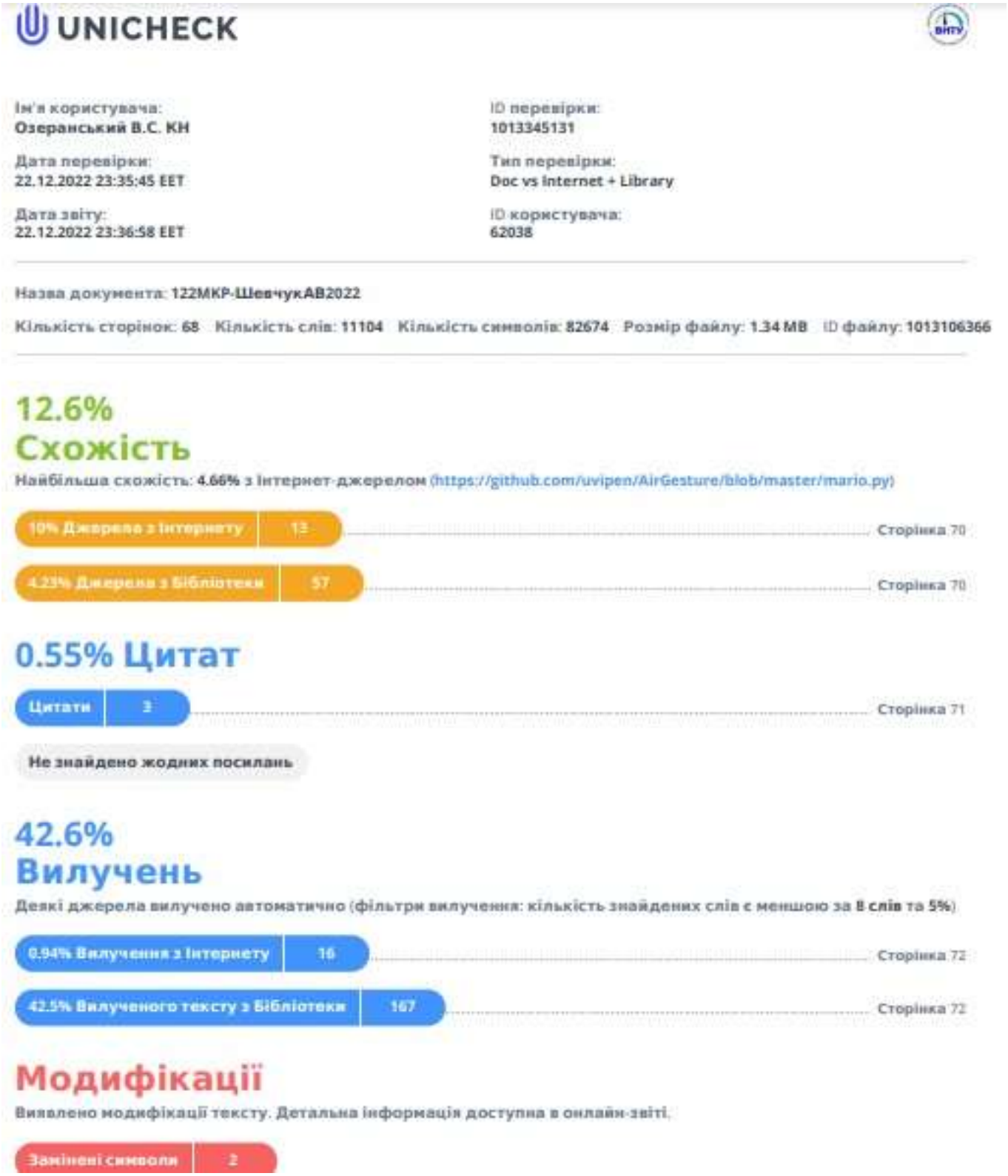
20. Дерево рішень [Електронний ресурс] – Режим доступу: https://studfile.net/preview/10096776/
21. Gradient descent [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/gradient-descent-3a7db7520711
22. Anti Aliasing [Електронний ресурс] – Режим доступу: https://vokigames.com/ua/borotba-iz-grafichnumu-nedolikamu-tekhnologiya-anti-aliasing-v-igrah/
23. Рекомендовані 8-розрядні формати YUV для відтворення відео [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/ru-ru/windows/win32/medfound/recommended-8-bit-yuv-formats-for-video-rendering
24. Recognition of overlapping elliptical objects in a binary image [Електронний ресурс] – Режим доступу: https://link.springer.com/article/10.1007/s10044-020-00951-z
25. Признаки Хаара [Електронний ресурс] – Режим доступу: https://studfile.net/preview/4326789/page:4/
26. Аналіз методу Віоли-Джонса для систем комп'ютерного зору [Електронний ресурс] – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/13162/11056
27. Герберт Шилдт: Java 8. Повне керівництво // Пер.з.англ.-М.:ООО "И.Д. Вильямс",2015.-31-32.с.
28. Марк Лутц: «Програмування на Python» // Пер. з англ. – СПб.: Діалектика, 2011.-С.41.
29. 4. C# і .NET [Електронний ресурс] − Режим доступу: https://metanit.com/sharp/tutorial/2.1.php
30. About XGBoost [Електронний ресурс] − Режим доступу: https://xgboost.ai/about
31. Implementing the DistBelief Deep Neural Network Training Framework with Akka [Електронний ресурс] − Режим доступу: http://alexminnaar.com/2015/09/06/DistBelief-with-Akka.html

32. Why TensorFlow always tops machine learning and artificial intelligence tool surveys [Електронний ресурс] – Режим доступу: https://hub.packtpub.com/tensorflow-always-tops-machine-learning-artificialintelligence-tool-surveys/
33. PyTorch vs TensorFlow — spotting the difference [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/pytorch-vs-tensorflow-spottingthe-difference-25c75777377b
34. NumPy [Електронний ресурс] – Режим доступу: https://numpy.org
35. Play classic games online, in a web browser! [Електронний ресурс] – Режим доступу: https://playclassic.games

ДОДАТКИ

Додаток А

Результат перевірки на плагіат в онлайн-системі UNICHECK



UNICHECK

Ім'я користувача:
Озеранський В.С. КН

ID перевірки:
1013345131

Дата перевірки:
22.12.2022 23:35:45 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
22.12.2022 23:36:58 EET

ID користувача:
62038

Назва документа: 122МКР-ШевчукАВ2022

Кількість сторінок: 68 Кількість слів: 11104 Кількість символів: 82674 Розмір файлу: 1.34 MB ID файлу: 1013106366

12.6%
Схожість

Найбільша схожість: 4.66% з інтернет-джерелом (https://github.com/uvipen/AirGesture/blob/master/mario.py)

10% Джерела з Інтернету 13 Сторінка 70

4.23% Джерела з Бібліотеки 57 Сторінка 70

0.55% Цитат

Цитати 3 Сторінка 71

Не знайдено жодних посилань

42.6%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 5%)

0.94% Вилучення з Інтернету 16 Сторінка 72

42.5% Вилученого тексту з Бібліотеки 167 Сторінка 72

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 2

Додаток Б

Лістинг програмного забезпечення

```
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from nes_py.wrappers import JoypadSpace
import gym
import cv2
from gym_chrome_dino.utils.wrappers import make_dino
from time import sleep
from src.config import HAND_GESTURES


def adjusted_detect_hand(img):

    hand_img = img.copy()

    hand_rect = hand_cascade.detectMultiScale(hand_img,
                                              scaleFactor =
1.2,

 minNeighbors = 5)

   lpalm_cascade =
 cv2.CascadeClassifier('classifiers/lpalm.xml')
   left_cascade =
 cv2.CascadeClassifier('classifiers/left.xml')
   fist_cascade =
 cv2.CascadeClassifier('classifiers/fist.xml')
   right_cascade =
 cv2.CascadeClassifier('classifiers/right.xml')
   rpalm_cascade =
 cv2.CascadeClassifier('classifiers/rpalm.xml')



   cap = cv2.VideoCapture(0)

   while 1:
 ret, img = cap.read()
```

```
   #gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


   lpalm = lpalm_cascade.detectMultiScale(img, 1.1, 3)
   left = left_cascade.detectMultiScale(img, 1.1, 3)
   fist = fist_cascade.detectMultiScale(img, 1.1, 3)
   right = right_cascade.detectMultiScale(img, 1.1, 3)
   rpalm = rpalm_cascade.detectMultiScale(img, 1.1, 3)



for (x,y,w,h) in lpalm:
   font = cv2.FONT_HERSHEY_SIMPLEX
   cv2.putText(img,'Left palm',(x-w,y-h), font, 0.5,
(11,255,255), 2, cv2.LINE_AA)
   sleep(1)


for (x,y,w,h) in left:
   font = cv2.FONT_HERSHEY_SIMPLEX
   cv2.putText(img,'Left Hand',(x-w,y-h), font, 0.5,
(11,255,255), 2, cv2.LINE_AA)
   sleep(1)



for (x,y,w,h) in fist:
   font = cv2.FONT_HERSHEY_SIMPLEX
   cv2.putText(img,'Fist',(x-w,y-h), font, 0.5, (11,255,255),
2, cv2.LINE_AA)
   sleep(1)


for (x,y,w,h) in right:
   font = cv2.FONT_HERSHEY_SIMPLEX
   cv2.putText(img,'Right Hand',(x-w,y-h), font, 0.5,
(11,255,255), 2, cv2.LINE_AA)
   sleep(1)


for (x,y,w,h) in rpalm:
   font = cv2.FONT_HERSHEY_SIMPLEX
```

```
    cv2.putText(img,'Right Palm',(x-w,y-h), font, 0.5,
 (11,255,255), 2, cv2.LINE_AA)
    sleep(1)


    cv2.imshow('img',img)
 k = cv2.waitKey(30) & 0xff
 if k == 27:
 break


    for (x, y, w, h) in hand_rect:
        cv2.rectangle(hand_img, (x, y),
                      (x + w, y + h), (255, 255, 255), 10)\

    return hand_img

inline __m128 ValidSqrt(__m128 value)
{
  return _mm_blendv_ps(_mm_set1_ps(1.0f), _mm_sqrt_ps(value),
_mm_cmpgt_ps(value, _mm_set1_ps(0.0f)));
}

inline __m128i Sum32ip(uint32_t * const ptr[4], size_t offset)
{
  __m128i s0 = _mm_loadu_si128((__m128i*)(ptr[0] + offset));
  __m128i s1 = _mm_loadu_si128((__m128i*)(ptr[1] + offset));
  __m128i s2 = _mm_loadu_si128((__m128i*)(ptr[2] + offset));
  __m128i s3 = _mm_loadu_si128((__m128i*)(ptr[3] + offset));
  return _mm_sub_epi32(_mm_sub_epi32(s0, s1), _mm_sub_epi32(s2,
s3));
}

inline __m128 Norm32fp(const HidHaarCascade & hid, size_t
offset)
{
  __m128 area = _mm_set1_ps(hid.windowArea);
  __m128 sum = _mm_cvtepi32_ps(Sum32ip(hid.p, offset));
  __m128 sqsum = _mm_cvtepi32_ps(Sum32ip(hid.pq, offset));
  return ValidSqrt(_mm_sub_ps(_mm_mul_ps(sqsum, area),
_mm_mul_ps(sum, sum)));
}
vid_placeholder = tf.placeholder(tf.float32,
```

```
shape=(batch_size * num_frames,
      image_size, image_size, 3))

    class GestureRecognition(rumps.App):
 def __init__(self, name):
 rumps.App.__init__(self, name)
    self.t = threading.Thread(target=self.detecting)
    self.t.start()
 print(self.resource_path("fist.xml"))
    newfile = open(self.resource_path("fist.xml"),"w+")
    for line in XMLDOCFIST:
 newfile.write(line)
 newfile.close()

    def resource_path(self, relative_path): # needed for
 bundling
    """ Get absolute path to resource, works for dev and for
 PyInstaller """
    base_path = getattr(sys, '_MEIPASS',
 os.path.dirname(os.path.abspath(__file__)))
    return os.path.join(base_path, relative_path)

    def detecting(self):
 self.video_capture = cv2.VideoCapture(0)
 while True:
    _, frame = self.video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    canvas = self.detect(gray, frame)
    if not self.video_capture:
    break
    video_capture.release()
    cv2.destroyAllWindows()

    def detect(self, gray, frame):
    fists = self.fist_cascade.detectMultiScale(gray, 1.3, 5)
    if len(fists) != 0:
    os.system(TOGGLE_OSASCRIPT.format(self.app_control,
 self.app_control, self.app_control, self.app_control))
    time.sleep(2)
    return frame

def is_in_triangle(point, triangle):
    # barycentric coordinate system
```

```
    x, y = point
    (xa, ya), (xb, yb), (xc, yc) = triangle
    a = ((yb - yc)*(x - xc) + (xc - xb)*(y - yc)) / ((yb -
yc)*(xa - xc) + (xc - xb)*(ya - yc))
    b = ((yc - ya) * (x - xc) + (xa - xc) * (y - yc)) / ((yb -
yc) * (xa - xc) + (xc - xb) * (ya - yc))
    c = 1 - a - b
    if 0 <= a <= 1 and 0 <= b <= 1 and 0 <= c <= 1:
        return True
    else:
        return False

def load_graph(path):
    detection_graph = tf.Graph()
    with detection_graph.as_default():
        graph_def = tf.GraphDef()
        with tf.gfile.GFile(path, 'rb') as fid:
            graph_def.ParseFromString(fid.read())
            tf.import_graph_def(graph_def, name='')
        sess = tf.Session(graph=detection_graph)
    return detection_graph, sess

def detect_hands(image, graph, sess):
    input_image = graph.get_tensor_by_name('image_tensor:0')
    detection_boxes =
graph.get_tensor_by_name('detection_boxes:0')
    detection_scores =
graph.get_tensor_by_name('detection_scores:0')
    detection_classes =
graph.get_tensor_by_name('detection_classes:0')
    image = image[None, :, :, :]
    boxes, scores, classes = sess.run([detection_boxes,
detection_scores, detection_classes],
                                      feed_dict={input_image:
image})
    return np.squeeze(boxes), np.squeeze(scores),
np.squeeze(classes)

def predict(boxes, scores, classes, threshold, width, height,
num_hands=2):
    count = 0
    results = {}
```

```
    for box, score, class_ in zip(boxes[:num_hands],
scores[:num_hands], classes[:num_hands]):
        if score > threshold:
            y_min = int(box[0] * height)
            x_min = int(box[1] * width)
            y_max = int(box[2] * height)
            x_max = int(box[3] * width)
            category = HAND_GESTURES[int(class_) - 1]
            results[count] = [x_min, x_max, y_min, y_max,
category]
            count += 1
    return results


def mario(v, lock):
    env = gym_super_mario_bros.make('SuperMarioBros-1-1-v0')
    env = JoypadSpace(env, COMPLEX_MOVEMENT)
    done = True
    while True:
        if done:
            env.reset()
            with lock:
                v.value = 0
        with lock:
            u = v.value
        _, _, done, _ = env.step(u)
        env.render()
        sleep(0.01)
tf.flags.DEFINE_integer("width", 1280, "Screen width")
tf.flags.DEFINE_integer("height", 720, "Screen height")
tf.flags.DEFINE_float("threshold", 0.6, "Threshold for score")
tf.flags.DEFINE_float("alpha", 0.3, "Transparent level")
tf.flags.DEFINE_string("pre_trained_model_path",
"src/pretrained_model.pb", "Path to pre-trained model")
FLAGS = tf.flags.FLAGS


def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
    mp = _mp.get_context("spawn")
```

```
    v = mp.Value('i', 0)
    lock = mp.Lock()
    process = mp.Process(target=mario, args=(v, lock))
    process.start()
    while True:
        key = cv2.waitKey(10)
        if key == ord("q"):
            break
        _, frame = cap.read()
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        boxes, scores, classes = detect_hands(frame, graph,
sess)
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        results = predict(boxes, scores, classes,
FLAGS.threshold, FLAGS.width, FLAGS.height)

        if len(results) == 1:
            x_min, x_max, y_min, y_max, category = results[0]
            x = int((x_min + x_max) / 2)
            y = int((y_min + y_max) / 2)
            cv2.circle(frame, (x, y), 5, RED, -1)

            if category == "Open" and x <= FLAGS.width / 3:
                action = 7  # Left jump
                text = "Jump left"
            elif category == "Closed" and x <= FLAGS.width / 3:
                action = 6  # Left
                text = "Run left"
            elif category == "Open" and FLAGS.width / 3 < x <= 2
* FLAGS.width / 3:
                action = 5  # Jump
                text = "Jump"
            elif category == "Closed" and FLAGS.width / 3 < x <=
2 * FLAGS.width / 3:
                action = 0  # Do nothing
                text = "Stay"
            elif category == "Open" and x > 2 * FLAGS.width / 3:
                action = 2  # Right jump
                text = "Jump right"
            elif category == "Closed" and x > 2 * FLAGS.width /
3:
                action = 1  # Right
```

```
                text = "Run right"
            else:
                action = 0
                text = "Stay"
            with lock:
                v.value = action
            cv2.putText(frame, "{}".format(text), (x_min, y_min
- 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)
        overlay = frame.copy()
        cv2.rectangle(overlay, (0, 0), (int(FLAGS.width / 3),
FLAGS.height), ORANGE, -1)
        cv2.rectangle(overlay, (int(2 * FLAGS.width / 3), 0),
(FLAGS.width, FLAGS.height), ORANGE, -1)
        cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 -
FLAGS.alpha, 0, frame)
        cv2.imshow('Detection', frame)

    cap.release()
    cv2.destroyAllWindows()
if __name__ == '__main__':
    main()

def dinosaur(v,lock):
    env = gym.make('ChromeDino-v0')
    env = make_dino(env, timer=True, frame_stack=True)
    done = True
    while True:
        if done:
            env.reset()
            with lock:
                v.value = 0
        with lock:
            u = v.value
        _, _, done, _ = env.step(u)
tf.flags.DEFINE_integer("width", 1280, "Screen width")
tf.flags.DEFINE_integer("height", 720, "Screen height")
tf.flags.DEFINE_float("threshold", 0.6, "Threshold for score")
tf.flags.DEFINE_float("alpha", 0.3, "Transparent level")
tf.flags.DEFINE_string("pre_trained_model_path",
"src/pretrained_model.pb", "Path to pre-trained model")

FLAGS = tf.flags.FLAGS
```

```
def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
    mp = _mp.get_context("spawn")
    v = mp.Value('i', 0)
    lock = mp.Lock()
    process = mp.Process(target=dinosaur, args=(v, lock))
    process.start()
    while True:
        key = cv2.waitKey(10)
        if key == ord("q"):
            break
        _, frame = cap.read()
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        boxes, scores, classes = detect_hands(frame, graph,
sess)
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        results = predict(boxes, scores, classes,
FLAGS.threshold, FLAGS.width, FLAGS.height)

        if len(results) == 1:
            x_min, x_max, y_min, y_max, category = results[0]
            x = int((x_min + x_max) / 2)
            y = int((y_min + y_max) / 2)
            cv2.circle(frame, (x, y), 5, RED, -1)
            if category == "Closed":
                action = 0  # Do nothing
                text = "Run"
            elif category == "Open" and y < FLAGS.height/2:
                action = 1 # Jump
                text = "Jump"
            elif category == "Open" and y > FLAGS.height/2:
                action = 2
                text = "Duck"
            else:
                action = 0
                text = "Run"
            with lock:
                v.value = action
```

```
            cv2.putText(frame, "{}".format(text), (x_min, y_min
- 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)
        overlay = frame.copy()
        cv2.rectangle(overlay, (0, 0), (FLAGS.width,
int(FLAGS.height / 2)), YELLOW, -1)
        cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 -
FLAGS.alpha, 0, frame)
        cv2.imshow('Detection', frame)
    cap.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```

Додаток В

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ КЕРУВАННЯ У КОМП'ЮТЕРНИХ ІГРАХ

Виконав: студент 2-го курсу,
групи 1КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Шевчук А.В.
(прізвище та ініціали)

Керівник: к. т. н., доцент каф. КН
_____ Арсенюк І.Р.
(прізвище та ініціали)
«15» 12 2022 р.

Рисунок В.1 – Структурна схема функціонування інформаційної технології розпізнавання жестів для керування у комп’ютерних іграх

Рисунок В.2 – Структурна схема алгоритму роботи інформаційної технології розпізнавання жестів для керування у комп’ютерних іграх

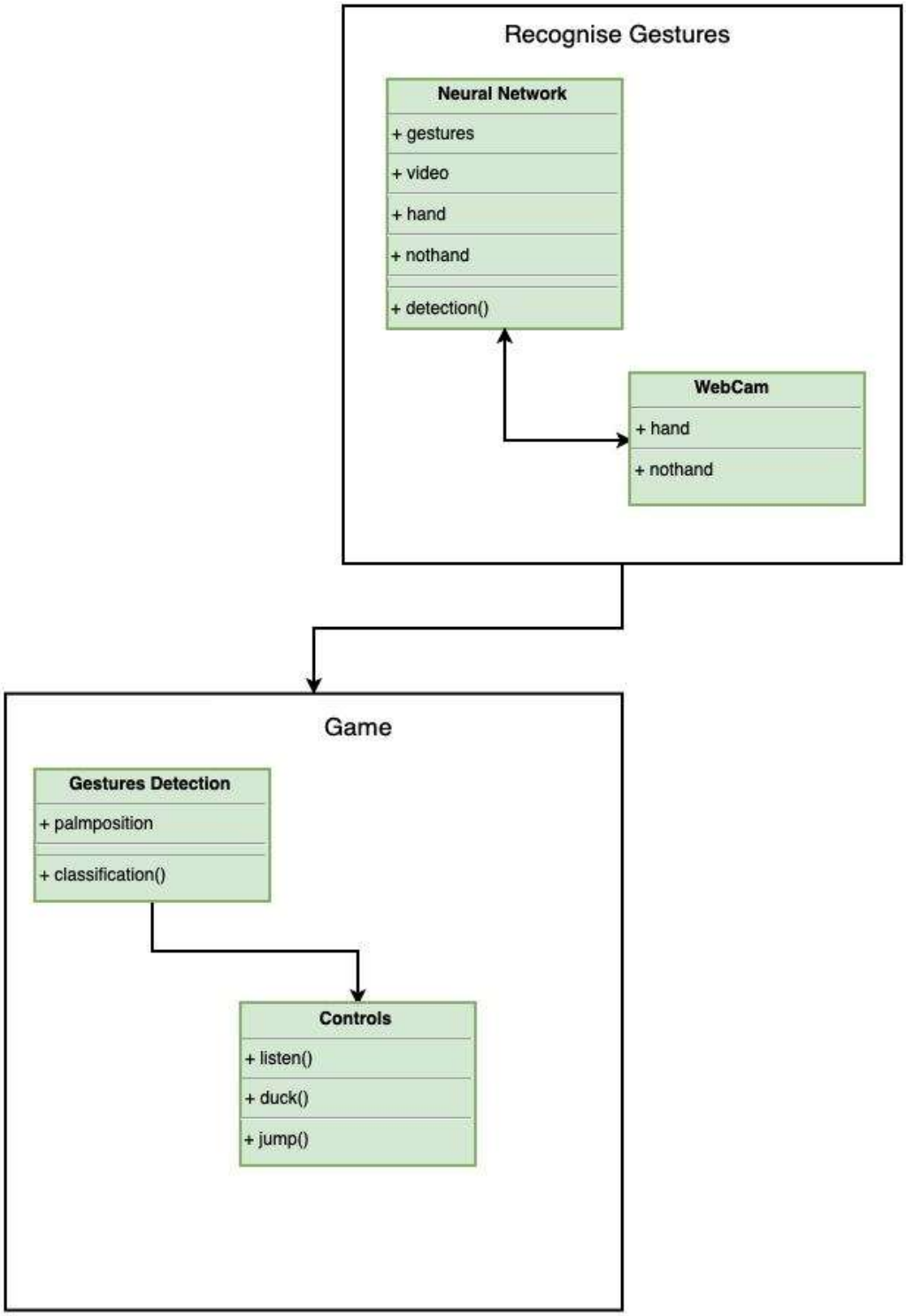


Рисунок В.3 – UML-діаграма компонентів системи розпізнавання жестів для керування у комп’ютерних іграх

Рисунок В.4 – Жестове керування у грі Dinosaur Game

Рисунок В.5 – Жестове керування у класичних іграх-платформерах